چگونه در C++ برنامه ننویسیم
یا
چرا ۲ + ۲ = ۵۹۸۶

نوشته: استیو اللاین

ترجمه: وحید مواجی

mavaji@gmail.com

مقدمه

رنج و زحمت، ابزاری شگفت برای یادگیری می باشد. طبیعت از طریق رنج می گوید: "این کار را انجام نده!"؛ اگر شما یک برنامه نویس باشید، سهم خود از رنج را برده اید. این امر معمولا حدود ۲ نیمه شب اتفاق می افتد وقتی که آخرین باگی را برای دو هفته شما را شکنجه می داد، می یابید.

این کتاب پر از برنامه های باگ دار است و به شما اجازه می دهد که از بدشانسی های بقیه برنامه نویس ها عبرت بگیرید. این شامل باگ هایی می شود که من یافته ام، باگ هایی که دوستانم یافته اند و باگ هایی که بقیه برنامه نویسان یافته اند. هر برنامه یک تجربه برای یادگیری است.

برنامه های ارائه شده در این جا، طوری طراحی شده اند که تا آنجا که ممکن است به برنامه های واقعی شبیه باشند. هر برنامه، یک کار ساده را انجام می دهد یا از یکی از ویژگی های زبان C++، استفاده می نماید. خبر بد این است که این برنامه ها کار نمی کنند. خبر خوب این است که همه آنها نسبتا کوچک هستند و لازم نیست که شما مثلا یک برنامه ۷۵۰۰۰۰ خطی را بالا و پایین کنید تا مشکل را بیابید.

برخی افراد معتقند که با تکنولوژی جدید کامپایلرها، اکثر این خطاها یافته می شوند. متأسفانه، خطاهای بسیاری وجود دارد که کامپایلرها نمی توانند آنها را بیابند.

برای مثال،از spell checker ها انتظار می رود که خطاهای املایی را حذف کنند. ولی می توانید در این جمله کاملا بی معنی خطای املایی پیدا کنید؟ "خروس های بویناک یا یک الهه فکر می کنند چون در طرف دیگر، این قالب ممکن است سوختی از پیکان ها باشد!" (یک spell checker نمی تواند هیچ خطای املایی در این جمله پیدا کند.

بنابراین سعی کنید خطاها را بیابید. اگر به مشکل برخوردید، ما راهنمایی هایی را فراهم کرده ایم که به شما کمک شود. همچنین پاسخها در انتهای کتاب می باشند. این امر در تضاد با زندگی واقعی است که هیچ راهنمایی در آن وجود ندارد و جوابی در انتهای کتابی برای آن نمی یابید.

این کتاب تقدیم می شود به همه برنامه نویسانی که روزهای متمادی با برنامه های پیچیده، باگ دار و پر از مشکل دست و پنجه نرم می کنند و مجبورند که راز معمای آنها را بگشایند.

فصل ۱: در آغاز[1]

در آغاز، ENIAC MARK I بود. روزی، اپراتوری متوجه شد که دستگاه درست کار نمی کند و فهمید که بیدی داخل دستگاه رفته و با برخورد به رله ها مرده است. او بید را بیرون انداخت و در گزارش کار خود نوشت " یک ساس در سیستم پیدا شد". و اینچنین اولین ساس (باگ) کامپیوتری کشف شد[2].

معرفی من از باگ های کامپیوتری، مدتها پس از آن واقعه انجام شد. من اولین برنامه ام را در سن ۱۱ سالگی نوشتم. طول آن فقط یک دستور اسمبلی بود. آن برنامه ۲ + ۲ را با هم جمع می کرد. نتیجه برابر ۲ می شد. طول آن برنامه فقط یک دستورالعمل بود و با این حال هم باگ داشت.

این فصل شامل یک سری مقدمات می شود: اولین باری که من تا ۲ نیمه شب بیدار ماندم تا یک باگ را پیدا کنم (برنامه ۳)، اولین سوالی که در اولین آزمون برنامه نویسی C طرح کردم (برنامه ۲) و البته اولین برنامه ای که در هر کتاب برنامه نویسی موجود می باشد: "Hello World".

قبل از اختراع ATM، شمل مجبور بودید به بانک بروید و بطور دستی، کارهای دریافت و پرداخت را انجام دهید. معمولا می توانستید از یکی از برگه های چاپ شده در دفترچه حساب خود استفاده کنید. شماره حساب شما با مرکب مغناطیسی در پایین برگه ها نوشته شده بود.

اگر برگه های شما تمام می شد، بانک یکی به شما می داد. در پایین آن، هیچ شماره ای نوشته نمی شد، لذا وقتی توسط دستگاه اتوماتیک بانک، پردازش می شد، دستگاه آنرا بیرون می داد و یک کارمند شماره حساب را بطور دستی در آن وارد می کرد.

یک کلاهبردار، برگه های "نوعی" خودش را چاپ کرد. آن شبیه برگه های "نوعی" معمولی بود بجز اینکه شماره حساب کلاهبردار با مرکب مغناطیسی در پایین آن نوشته شده بود. او سپس به بانک رفت و آن برگه ها در سبد برگه های "نوعی" انداخت.

کلاهبرداری بدین صورت بود: یک مشتری وارد بانک شد تا کار بانکی انجام دهد و یکی از آن برگه های دستکاری شده را برداشت. او آن برگه را پر کرد و پول پرداخت کرد. از آنجا که برگه، شماره حساب داشت، کامپیوتر به صورت اتوماتیک آنرا پردازش کرد و پولی به حساب نوشته شده در پایین برگه واریز کرد. آنچه که به آن توجه نشد، شماره حسابی بود که بطور دستی روی برگه نوشته شده بود. بعبارت دیگر، کلاهبردار ما داشت پولها را می دزدید.

کارگاه مسئول این قضیه گیج شده بود. پول ها ناپدید می شدند و کسی نمی دانست چگونه. او کار را به پولهایی که در بانک پرداخت می شدند محدود کرد. او تصمیم گرفت که تعداد زیادی پرداخت انجام دهد و ببیند که چه اتفاقی می افتد. چون او از جیب خودش داشت خرج می کرد، پولهایی که پرداخت می کرد، بسیار کم بود. بسیار بسیار کم. در حقیقت هر کدام ۶ سنت بودند.

کارآگاه یک هفته را به این کار گذراند. به بانک می رفت، یک برگه پر می کرد، در صف می ایستاد، ۶ سنت پرداخت می کرد، یک برگه جدید پر می کرد، در صف می ایستاد، ۶ سنت پرداخت می کرد و الخ. کارمندان فکر می کردند که او دیوانه شده است. یک روز، یکی از پرداخت هایش ناپدید شد. او بانک را مجبور کرد که بررسی کند که آیا کس دیگری در آنروز، یک پرداخت ۶ سنتی داشته یا نه. یکی داشت و اینگونه کلاهبردار به دام افتاد.

[1] In the Beginning

[2] با این که افراد معتقند که این واقعه، اولین کاربرد کلمه*باگ* در مورد ماشین های محاسباتی بود، ولی اینگونه نبود. اصطلاح باگ برای مدت مدیدی قبل از آن به همه گونه اشکالات ماشین افزار اطلاق می شود. بهر حال چرا یک داستان خوب را با واقعیت خراب کنیم؟

برنامه ۱: Hello World

به نظر می رسد که "Hello World"، اولین برنامه در هر کتاب برنامه نویسی باشد و فرقی نمی کند. ولی این یکی مشکل دارد. چگونه می توان چیزی به سادگی یک "Hello World" را خراب کرد؟ نگاه کنید:

```
1 /*************************************************
2 * The “standard” hello world program. *
3 *************************************************/
4 #include <iostream>
5
6 void main(void)
7 {
8         std::cout << “Hello world!\n”;
9 }
```

(راهنمایی ۲۲۸، جواب ۶)

کاربر: امروز نمی توانم وارد سیستم شود. مودم وصل نمی شود.
پشتیبان: مودم خود را ببین و بگو کدام چراغها روشن هستند.
کاربر: نمی توانم این کار را بکنم.
پشتیبان: خب، من نمی توانم کمکی به حل مشکل شما بکنم مگر اینکه بگویید آنجا چه خبر است. نمی توانی به مودم نگاه کنی و وضعیت آنرا بگویی؟
کاربر: نه، نمی توانم.
پشتیبان: چرا نه؟
کاربر: مودم، پایین در سرداب است.
پشتیبان: خب چرا نمی روی پایین و به آن نگاه کنی؟
کاربر: شوخی می کنی؟ شش فوت آب آن زیر است!
پشتیبان: کامپیوترها زیر آب کار نمی کنند.
کاربر(با شگفتی): واقعا؟

برنامه ۲: مشکل استاد[1]

من برنامه نویسی C درس می دهم. این اولین سوال اولین آزمونی است که برگزار کرده ام. ایده کار ساده بود: می خواستم ببینم آیا دانش آموزان فرق بین متغیر **automatic**

16 int i = 0;

و متغیر **static**

26 static int i = 0;

را می دانند یا نه. با این حال بعد از آزمون، مجبور شدم مسأله شرم آوری را بپذیرم: اگر خودم در این آزمون شرکت می کردم، به این سوال، اشتباه جواب می دادم. لذا مجبور شدم به دانش آموزان بگویم: "برای اینکه نمره کامل سوال ۱ را بگیرید، دو راه وجود دارد. راه اول این است که جواب درست داده باشید و راه دوم این است که جوابی را که من فکر می کردم درست است، داده باشید".

بنابراین، جواب درست کدام است؟

```
1 /*************************************************
2 * Test question: *
3 *        What does the following program print? *
4 * *
5 * Note: The question is designed to tell if *
6 * the student knows the difference between *
7 * automatic and static variables. *
8 *************************************************/
9 #include <stdio.h>
10 /*************************************************
11 * first -- Demonstration of automatic *
12 * variables. *
13 *************************************************/
14 int first(void)
15 {
16        int i = 0; // Demonstration variable
17
18         return (i++);
19 }
20 /*************************************************
21 * second -- Demonstration of a static *
22 * variable. *
23 *************************************************/
24 int second(void)
25 {
26        static int i = 0; // Demonstration variable
27
28        return (i++);
29 }
30
31 int main()
32 {
33        int counter; // Call counter
34
35        for (counter = 0; counter < 3; counter++)
36                printf("First %d\n", first());
37
38        for (counter = 0; counter < 3; counter++)
39                printf("Second %d\n", second());
40
```

[1] Teacher's Problem

```
41        return (o);
42 }
```

(راهنمایی ۱۳۹، جواب ۱۰۲)

کلیسایی، تازه اولین کامپیوتر خود را خریده بود و کارکنان آن مشغول یادگیری روش استفاده از آن بودند. منشی کلیسا تصمیم گرفت متنی را تنظیم کند تا در مراسم ترحیم استفاده شود. جایی که اسم شخص مورد نظر باید تغییر می کرد، کلمه <name> بود. وقتی مراسم ترحیمیم قرار بود انجام شود، او این کلمه را با اسم واقعی شخص عوض می کرد.

روزی، دو مراسم ترحیم بود. اولی برای بانویی به اسم مریم و دومی برای شخصی به اسم ادنا. لذا منشی، هرجا که <name> بود را با "مریم" عوض کرد. تا این جا همه چیز به خوبی پیش رفت. سپس او برای دومین مراسم ترحیم، تمام "مریم" ها را با "ادنا" عوض کرد. این، یک اشتباه بود.

کشیش را تصور کنید که قسمتی از "اعمال رسولان" را می خواند و می بیند که نوشته "زاده شد از ادنای باکره".

برنامه ۳: شگفتی در صبح زود[1]

این برنامه توسط یکی از دوستانم نوشته شده وقتی که هر دو در دانشگاه بودیم. تمرین خانه این بود که یک روتین ضرب ماتریسی بنویسیم. هرچند خود تابع باید به زبان اسمبلی نوشته می شد. برای این سرعت اجرای آنرا تا آنجا که می توانیم افزایش دهیم، دوستم باید از الگوریتمی که من طراحی کرده بودم و ماتریس را بصورت بردار در می آورد، استفاده می کرد.

برای آزمایش سیستم، او یک تابع تست در SAIL نوشت[2]. وقتی برنامه را تست کردیم، جوابهای غلطی بدست آوردیم. هر دوی ما، خط به خط برنامه را از ۸ بعد از ظهر تا ۲ نیمه شب موشکافی کردیم. وقتی نهایتا خطا را پیدا کردیم، از اینکه مرتکب چنین اشتباه احمقانه ای شده بودیم، به شدت خندیدیم.

برنامه زیر یک نسخه ساده شده از آن کد است. تمام این برنامه به زبان C نوشته شده است و از الگوریتم ساده تری برای ضرب استفاده می کند. ولی باگ اولیه کماکان وجود دارد. مشکل کجاست؟

```
1 /*************************************************
2 * matrix-test -- Test matrix multiply *
3 *************************************************/
4 #include <stdio.h>
5
6 /*************************************************
7 * matrix_multiply -- Multiple two matrixes *
8 *************************************************/
9 static void matrix_multiply(
10        int result[3][3], /* The result */
11        int matrixl[3][3],/* One multiplicand */
12        int matrix2[3][3] /* The other multiplicand */
13 )
14 {
15        /* Index into the elements of the matrix */
16        int row, col, element;
17
18         for(row = 0; row < 3; ++row)
19        {
20                for(col = 0; col < 3; ++col)
21                {
22                        result[row][col] = 0;
23                        for(element = 0; element < 3; ++element)
24                        {
25                                result[row][col] +=
26                                matrix1[row][element] *
27                                matrix2[element][col];
28                        }
29                }
32        }
33 }
34
35 /*************************************************
36 * matrix_print -- Output the matrix *
37 *************************************************/
38 static void matrix_print(
39        int matrix[3][3] /* The matrix to print */
40 )
41 {
```

[1] Early Morning Surprise

[2] SAIL یک زبان قدیمی برای برنامه نویسی سیستم PDP-10 بود. دیباگر آن BAIL نام داشت. بعدها یک نسخه مستقل از ماشین این زبان ابداع گردید که MAIN SAIL نام داشت. این زبان چندین سال قبل از C وجود داشت.

```
42          int row, col; /* Index into the matrix */
43
44          for (row = 0; row < 3; ++row)
45          {
46                      for (col = 0; col < 3; ++col)
47                      {
48                                  printf("%o\t", matrix[row][col]);
49                      }
50                      printf("\n");
51          }
52 }
53
54 int main(void)
55 {
56          /* One matrix for multiplication */
57          int matrix_a[3][3] = {
58                      {45, 82, 26},
59                      {32, 11, 13},
60                      {89, 81, 25}
61          };
62          /* Another matrix for multiplication */
63          int matrix_b[3][3] = {
64                      {32, 43, 50},
65                      {33, 40, 52},
66                      {20, 12, 32}
67          };
68          /* Place to put result */
69          int result[3][3];
70
71          matrix_multiply(result, matrix_a, matrix_b);
72          matrix_print(result);
73          return (o);
74 }
```

(راهنمایی ۳۴، جواب ۵۳)

فصل ۲: خشت اول چون نهد معمار کج[1]

همه ما زمانی برنامه نویسان مبتدی بودیم. آن موقع ممکن بود ساعت ها جان بکنیم تا بتوانیم ساده ترین برنامه را کامپایل کنیم. ولی آن موقع جوان و جاهل بودیم و اشتباه های احمقانه ای مرتکب می شدیم. حالا برنامه نویسان حرفه ای هستیم و اشتباه های احمقانه نمی کنیم بلکه اشتباه های زیرکانه می کنیم (فقط اسم آنرا گذاشته ایم "خطاهای حرفه ای").

در این فصل، تعدادی برنامه ارائه می شود که بدین منظور طراحی گردیده اند تا اشتباه های اولیه برنامه نویسی را گوشزد کنند. بنابراین این تجربیات را دوباره مرور کنید تا از ذهنتان پاک نشود.

برنامه ۴: مشکل نوعی مقداردهی اولیه[2]

یک مسأله کلاسیک ریاضی، جمع اعداد ۱ تا ۱۰۰ است. ولی به نظر می رسد که این برنامه این کار را درست انجام نمی دهد:

```
1 /************************************************
2 * A program to sum the numbers from 1 to 100 *
3 * using a brute force algorithm. *
4 ************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         int sum; // The running sum
10        int count; // The current number
11
12        for (count = 1; count <= 100; ++count)
13                sum += count;
14
15        std::cout <<
16                "The sum of the numbers " <<
17                "between 1 and 100 is " <<
18                sum << '\n';
19        return (0);
20 }
```

(راهنمایی ۱۱۶، جواب ۵۱)

یک کارخانه مونتاژ لوازم الکترونیکی با مشکل دله دزدی مواجه شد. هزاران قطعه الکترونیکی ناپدید می شدند. کارخانه مقررات امنیتی زیادی را وضع کرد، ولی کمبودها کماکان ادامه داشت. این قطعات کجا ممکن بود رفته باشند؟

بالاخره یک سرایدار پرده از راز معما گشود. او آن بالا داشت لامپی را عوض می کرد که سه لانه پرنده را پیدا کرد. پرنده ها قطعات را از کف کارخانه برمی داشتند و از آنها در ساخت لانه هایشان استفاده می کردند. طبق برآوردها، هر کدام از لانه ها، ده هزار دلار می ارزیدند.

[1] Starting out on the Wrong Foot
[2] Typical Initial Rroblem

برنامه ۵: اولین خطاها[1]

هر برنامه نویس مبتدی، با یادگرفتن در مورد عبارات ساده و نمایش آنها شروع می کند. برنامه زیر بسیار ساده است. مشکل چیست؟

```
1 /*************************************************
2 * A program to answer the question of a five *
3 * year old: *
4 * "What is 2 + 2?" *
5 *************************************************/
6 #include <iostream>
7
8 int main()
9 {
10        int result; // Result of the addition
11
12        result = 2+2;
13        std::cout << "The answer is " << result;
14        return (0);
15 }
```

(راهنمایی ۲۵۱، جواب ۴۳)

یک برنامه نویس زیرک، راهی برای سرقت از بانک پیدا کرد.او از هر پرداخت کننده ۱,۲ سنت می دزدید. وقتی بانک ها بهره مرکب را محاسبه می کنند، نتیجه همیشه یک عدد سرراست نیست. مثلا بهره می تواند ۳,۲ یا ۸,۶ سنت باشد. بانک ها معمولا این عدد را گرد می کنند بنابراین ۳,۲ به ۳ و ۸,۶ به ۹ تبدیل می شود. نتیجه این است که نصف مواقع، عدد به بالا و نصف مواقع عدد به پایین گرد می شود. پس حسابها درست در می آید.

یک برنامه نویس نادرست، الگوریتم را طوری تغییر داد که همیشه عدد را قطع کند. لذا ۳,۲ به ۳ و ۸,۶ به ۸ تبدیل می شود. این کار مقدارزیادی از سنت ها را باقی می گذاشت. برنامه نویس، این مقادیر را جمع می کرد و آنرا به حساب آخرین نفر در لیست حسابها می ریخت. از آنجا که او حسابی به اسم ZZYMOCK باز کرده بود، این پولها به حساب او می رفت.

دزد ما خیلی زرنگ بود. از هر کس کمتر از یک سنت دزدید و هیچ کس متوجه نشد. علاوه براین چه کسی مبلغ بهره خود را تا آخرین رقم اعشار چک می کند؟ اصلا چند نفر، مبلغ بهره خود را چک می کنند؟

ولی گیر افتاد. ZZYSKI یک حساب باز کرد. حالا اسم او در انتهای لیست بود. وقتی که او اولین بار موجودی گرفت، تقریبا شگفت زده بود که چگونه با ۲۰۰ دلار حساب، بهره ای به اندازه ۳۸۲۳۸,۸۳ دلار گرفته است.

[1] First Errors

برنامه ۶: بر سر *فاصله* چه آمد؟[1]

این یک برنامه کوتاه آزمایشی است که توسط شخصی در اولین روزهای برنامه نویسی اش نوشته شده است. این برنامه برای نمایش یک جواب ساده طراحی شده است. ولی کارها درست پیش نمی روند.

```
1 /*************************************************
2 * Double a number. *
3 *************************************************/
4 #include <iostream>
5
6 int main(void)
7 {
8         int number; // A number to double
9
10        std::cout << "Enter a number:";
11        std::cin >> number;
12
13         std::cout << "Twice" << number << "is" <<
14                 (number * 2) << '\n';
15        return (0);
16 }
```

(راهنمایی ۲۴۷، جواب ۲۳)

من مدتی برنامه نویسی تدریس می کردم. در آن زمان چیز زیادی درباره تدریس نمی دانستم و تعیین میزان تکلیف برای دانش آموزان برایم سخت بود. یک بار توسط پلیس فورت ورث متوقف شدم چون تکلیف هایم خیلی سخت بود. ماجرایی واقعی.

داشتم در خیابانهای فورت ورث رانندگی می کردم و پشت یک چراغ قرمز توقف کردم. یک ماشین پلیس کنار من ایستاد. من به افسر نگاه کردم. او لحظه ای به من نگاه کرد و اشاره کرد که شیشه ماشینم را پایین بیاورم. اقرار می کنم که کمی نگران بودم. تازه، من داشتم با یک شورولت ۵۸ تعمیر نشده که اگزوز آن سه بار افتاده بود حرکت می کردم.

شیشه را پایین آوردم و او به من فریاد زد که "استیو، تکلیف های این هفته ات خیلی مشکل اند".

آن موقعی بود که فهمیدم یکی از دانش آموزانم برای اداره پلیس فورت ورث کار می کرده است. نیازی به گفتن نیست که من یک هفته اضافه به دانش آموزان وقت دادم تا تکلیف های خود را تحویل دهند.

[1] Gotta Have My Space

برنامه ۷: مجذور نادرست[1]

این یک برنامه کوتاه برای محاسبه و نمایش مجذور اعداد ۱ تا پنج است. به اندازه کافی ساده است، پس کجای آن غلط است؟

```
1 /************************************************
2 * squares -- Print the squares of the numbers *
3 * from 1 to 5. *
4 ************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         // An array for the squares
10        int array[5];
11
12        int i; // Index into the array
13
14        for (i = 1; i <= 5; ++i) {
15                array[i] = i*i;
16        }
17
18        for (i = 1; i <= 5; ++i) {
19                std::cout << i << " squared is " <<
20                        array[i] << '\n';
21        }
22        return (0);
23 }
```

(راهنمایی ۱۰۳، جواب ۹۰)

کنار اتاق کامپیوتر یک شرکت آمریکایی پیدا شد:

ACHTUNG! ALLES LOOKENSPEEPERS!

Das computermachine ist nicht fuer gefingerpoken und mittengrabben. Ist easy schnappen der springenwerk, blowenfusen und poppencorken mit spitzensparken. Ist nicht fuer gewerken bei das dumpkopfen. Das rubber-necken sichtseeren keepen das cotten-pickenen hans in das pockets muss; relaxen und watchen das blinkenlichten.

[1] The Crooked Square

برنامه ۸: کاراکتر سرگشته[1]

برنامه نویس مبتدی تصمیم گرفته تا دستور **if** را با متغیرهای **char** استفاده کند. برنامه زیر، ساده، واضح و غلط است!

```
1 /************************************************
2 * Check the flag. *
3 ************************************************/
4 #include <iostream>
5
6 int main()
7 {
8         char ch; // The flag
9
10        ch = 0xFF; // Set the flag
11
12        // Check the flag
13        if (ch == 0xFF)
14                std::cout << "Success\n";
15        else
16                std::cout << "Fails\n";
17
18 return (0);
19 }
```

(راهنمایی ۱۳۱، جواب ۸)

کنار اتاق کامپیوتر یک شرکت آلمانی پیدا شد:

ATTENTION

This room is fullfilled mit special electronische equippment. Fingergrabbing and pressing the cnoeppkes from the computers is allowed for die experts only! So all the "lefthanders" stay away and do not disturben the brainstorming von here working intelligencies. Otherwise you will be out thrown and kicked anderswhere! Also: Please keep still and only watchen astaunished the blinkenlights.

[1] Mad Character

برنامه ۹: بدون شرح[1]

این برنامه، مساحت یک مثلث را محاسبه می کند. فرمول آن ساده است و مشخص است که همه چیز کار می کند ولی یک اشکال شگفت آور در این کد وجود دارد:

```
1 /*************************************************
2 * triangle -- Compute the area of a triangle *
3 *************************************************/
4 #include <iostream>
5 int main()
6 {
7         int base = 0; /* Base of the triangle */
8         int height = 0; /* Height of the triangle */
9
10        base = 5; /* Set the base of the triangle
11        height = 2; /* Set the height */
12
13        // Area of the triangle
14        int area = (base * height) / 2;
15
16        std::cout << "The area is " <<
17                area << std::endl;
18        return (0);
19 }
```

(راهنمایی ۴۱، جواب ۶۲)

یک مدیر سیستم، با روتر شبکه مشکلات زیادی داشت. شماره خطاهای عجیب و غریبی مثل "E6" و "B2" روی صفحه نمایش دیده می شدند. او با سازنده تماس گرفت و به قسمت خدمات پس از فروش وصل شد.

مدیر سیستم: می توانید به من بگویید کد E6 یعنی چه؟
تکنیسین: خط ارتباطی ۶ اتصال کوتاه کرده.

- این مطلب کجا نوشته شده است؟
- در دفترچه راهنمای فنی.
- ما مشکلات زیادی اینجا داریم، می توانید یک نسخه از آن راهنما را برای من فکس کنید؟
- (با اکراه) خب، باشد. ولی این تنها نسخه ای است که دارم. قول بدهید که آنرا دوباره به من فکس می کنید.

[1] No Comment

برنامه ۱۰: تقسیم نه چندان بزرگ[1]

این برنامه ساده ای است که نشان می دهد چند رقم بامعنی برای اعداد اعشاری استفاده می شود. ایده آن ساده است: یک کسر با دور گردش مثل 0.3333333 = 1/3 را در نظر بگیر، آنرا نمایش بده و ببین چند رقم آن نمایش داده می شود.

با این حال، نتایج برنامه، برنامه نویس را گیج کرد. او می دانست که کامپیوتر نمی تواند اینقدر احمق باشد، پس چه اتفاقی افتاده؟

```
1 /************************************************
2 * divide -- Program to figure out how many *
3 * digits are printed in floating point *
4 * by print 1/3 or 0.333333. *
5 ************************************************/
6 #include <iostream>
7
8 int main()
9 {
10 float result; // Result of the divide
11
12 result = 1/3; // Assign result something
13
14 std::cout << "Result is " << result << '\n';
15 return (0);
16 }
```

(راهنمایی ۲۹۲، جواب ۲۷)

هواشناسی باید دریک کامپیوتر اداره هواشناسی، مقدار باران را بر حسب اینچ وارد می کرد. افراد آنجا عادت کرده بودند که با صدم اینچ کار کنند، لذا وقتی از آنها پرسیده می شد که امروز چقدر باران آمده، جواب ۵۰ به معنی ۵۰/۱۰۰ اینچ یا نصف اینچ می بود.

با این حال برای وارد کردن این در کامپیوتر باید نوشته می شد "0.50". یکی از افراد این مطلب را فراموش کرد و میزان بارش باران را بصورت "50" وارد کرد. حالا ۵۰ اینچ، باران *زیادی* است. میزان فوق العاده ای باران است. کامپیوتر خطا گرفت و پیغام مناسب را نمایش داد:
یک کشتی بساز و از هر کدام از جانداران جفتی را بردار...

[1] The Not-So-Great-Divide

برنامه ۱۱: دو فایل، خیلی زیاد است[1]

این هم روش دیگری برای انجام دادن "Hello World!" و اشتباه در آن است. مشکل کجاست؟

File: sub.cpp
```
1 // The string to print
2 char str[] = "Hello World!\n";
```

File: main.cpp
```
1 /************************************************
2 * print string -- Print a simple string. *
3 ************************************************/
4 #include <iostream>
5
6 extern char *str; // The string to print
7
8 int main()
9 {
10        std::cout << str << std::endl;
11        return (0);
12 }
```

(راهنمایی ۲۶۹، جواب ۷)

برنامه نویسی که من او را می شناسم فکر می کرد که راهی را یافته است که چگونه هیچ وقت کارت پارک تهیه نکند. سه گزینه او برای پلاک ماشین اینها بودند: ۱) 0O0O0O، ۲) O0O0O0، و ۳) I1I1I1. او تصور می کرد که اگر مأمور پلیسی ماشین را ببیند، حرف O و رقم 0 بسیار شبیه هم می باشند و تقریبا غیرممکن است که شماره پلاک را درست یادداشت کند.

متأسفانه نقشه او نگرفت. مأمور راهنمایی رانندگی که پلاک را صادر می کرد سردرگم شد و شماره پلاک را بصورت OOOOOO صادر کرد.

[1] Two Files Is Too Many

برنامه ۱۲: زودباش و صبر کن[1]

کدی که بر اساس آن، این برنامه نوشته شده است، توسط یک برنامه نویس سیستم در شرکتی که مدتها پیش در آن کار می کردم، نوشته شده است.

قرار بود این برنامه روی یک خط سریال، داده بفرستد. با اینکه خط سریال می توانست تا ۹۶۰ کاراکتر در ثانیه را رد و بدل کند، ما فقط می توانستیم ۳۰۰ کاراکتر در ثانیه داشته باشیم. چـــــــــــــرا؟

```
1 /************************************************
2 * send_file -- Send a file to a remote link *
3 * (Stripped down for this example.) *
4 ************************************************/
5 #include <iostream>
6 #include <fstream>
7 #include <stdlib.h>
8
9 // Size of a block
10 const int BLOCK_SIZE = 256;
11
12 /************************************************
13 * send_block -- Send a block to the output port*
14 ************************************************/
15 void send_block(
16         std::istream &in_file, // The file to read
17         std::ostream &serial_out // The file to write
18 )
19 {
20         int i; // Character counter
21
22         for (i = 0; i < BLOCK_SIZE; ++i) {
23                 int ch; // Character to copy
24
25                 ch = in_file.get();
26                 serial_out.put(ch);
27                 serial_out.flush();
28         }
29 }
30
31 int main()
32 {
33         // The input file
34         std::ifstream in_file("file.in");
35
36         // The output device (faked)
37         std::ofstream out_file("/dev/null");
38
39         if (in_file.bad())
40         {
41                 std::cerr <<
42                         "Error: Unable to open input file\n";
43                 exit (8);
44         }
45
46         if (out_file.bad())
47         {
48                 std::cerr <<
49                         "Error: Unable to open output file\n";
```

[1] Hurry Up and Wait

```
50                  exit (8);
51          }
52
53          while (! in_file.eof())
54          {
55                  // The original program output
56                  // a block header here
57                  send_block(in_file, out_file);
58                  // The original program output a block
59                  // trailer here. It also checked for
60                  // a response and resent the block
61                  // on error
62          }
63          return (0);
64 }
```

(راهنمایی ۱۸۳، جواب ۶۵)

یک مدیر سیستم عادت دارد که دو هفته قبل ازاینکه سیستم ها را ارتقا دهد، اعلام می کند که کار ارتقا انجام شده است. نوعا اعتراض های عجولانه ای مانند "نرم افزار من از کار افتاده است و این نتیجه ارتقا شما است" در روز اعلان وجود خواهد داشت. مدیر می داند که این امر بدلیل ارتقا نیست چون واقعا آنرا انجام نداده است.

وقتی که او واقعا عمل ارتقا را انجام می دهد (بطور مخفیانه) هر اعتراضی که بعد از آن وجود داشته باشد احتمالا برحق است.

اپراتورهای آماتور رادیو از این ترفند استفاده می کنند. آنها یک برج رادیویی جدید نصب می کنند و آنرا برای چند هفته قطع نگه می دارند. این کار به همسایه ها دو هفته فرصت می دهد تا به تداخل تلویزیون بخاطر وجود آنتن جدید اعتراض کنند.

برنامه ۱۳: این برنامه خیلی اگر دارد[1]

چرا این برنامه برای بعضی مبالغ درست کار نمی کند؟ همچنین این برنامه علاوه بر مشکلی که قرار بود آنرا نشان دهد، خطایی هم دارد. مشکل دیگر کجاست؟

```
1 /*************************************************
2 * Billing -- Print out how much we owe *
3 * customers or they owe us. *
4 *************************************************/
5 #include <iostream>
6
7 // Number of pennies in a dollar
8 const int DOLLAR = 100;
9
10 /*************************************************
11 * billing -- do the billing. *
12 * If the customer owes us money *
13 * -- output debt. *
14 * If we owe more than $100 *
15 * -- output credit. *
16 * Between $0 and $100 just ignore the *
17 * account. *
18 *************************************************/
19 int billing(
20 // Current balance (in cents)
21         const int balance
22 ) {
23         if (balance < 0)
24                 if (balance < - (100*DOLLAR))
25                         std::cout << "Credit " << -balance << endl;
26         else
27                 std::cout << "Debt " << balance << endl;
28
29         return (0);
30 }
31
32 int main()
33 {
34         /* Test code */
35         billing(50);
36         billing(-10);
37         return (0);
38 }
```

(راهنمایی ۴۴، جواب ۳۱)

[1] This Program Is a Little Iffy

برنامه ۱۴: برنامه نویسی فریب آمیز[1]

برنامه نویس می داند که شیفت دادن به چپ مانند ضرب در توانی از ۲ است. بعبارت دیگر:

x << 1 برابر است با x * 2 ($2 = 2^1$)
x << 2 برابر است با x * 4 ($4 = 2^2$)
x << 3 برابر است با x * 8 ($8 = 2^3$)

برنامه نویس از این ترفند برای انجام یک محاسبه سریع استفاده می کند ولی یک چیز اشتباه است:

```
1 /********** ********** ********** **********
2 * Simple syntax testing. *
3 ************************************************/
4 #include <iostream>
5
6 int main(void)
7 {
8         int x,y; // Two numbers
9
10        x = 1;
11
12        y = x<<2 + 1; // x<<2 = 4 so y = 4+1 = 5
13        std::cout << "Y=" << y << std::endl;
14        return (0);
15 }
```

(راهنمایی ۲۶۶، جواب ۴۹)

یک هکر مأمور شد تا برنامه ای بنویسد که یک ماشین حساب چهار کاره را شبیه سازی کند. این برنامه باید عمل جمع، تفریق، ضرب و تقسیم را انجام می داد. با این حال مشخص نشده بود که چه نوع عددی استفاده باید شود. بنابراین برنامه هکر با اعداد رومی کار می کرد(IV + III = VII). به یک دفترچه راهنمای کاربر نیز نیاز بود ولی زبان آنهم مشخص نشده بود. بنابراین برنامه نویس یک راهنمای مفصل به زبان لاتین تهیه کرد.

[1] Shifty Programming

برنامه ۱۵: خاموش[1]

برنامه زیر به منظور تشخیص اینکه یک کلمه، کلمه کلیدی است یا نه، طراحی شده است. پس چرا کار نمی کند؟

```
1 /*************************************************
2 * test the keyword finding function: "keyword" *
3 *************************************************/
4 #include <cstring>
5 #include <iostream>
6
7 /*************************************************
8 * keyword -- return true if a keyword found *
9 *************************************************/
10 bool keyword(
11         const char word[] // The work to look for
12 )
13 {
14         // A set of keywords
15         static const char *key_list[] = {
16                 "bool",
17                 "int",
18                 "const",
19                 NULL
20         };
21         int i; // Index into the list
22
23         // Look for the keyword
24         for (i = 0; key_list[i] != 0; ++i) {
25                 if (std::strcmp(word, key_list[i]))
26                         return (true);
27         }
28         return (false);
29 }
30 int main()
31 {
32         std::cout << "keyword(bool) = " <<
33                 keyword("bool") << '\n';
34
35         std::cout << "keyword(sam) = " <<
36                 keyword("sam") << '\n';
37         return (0);
38 }
```

(راهنمایی ۲۹۴، جواب ۷۶)

[1] Wordless

برنامه ۱۶: آهسته و پیوسته[1]

چرا این برنامه اینقدر کند است؟ روی سیستم من ۱ دقیقه و ۳۴ ثانیه طول می کشد تا فایل را کپی کند، درحالیکه دستور **cp** لینوکس همین کار را در کمتر از نیم ثانیه انجام می دهد. چه کاری می توان کرد تا برنامه سریعتر شود؟

```
1 /************************************************
2 * copy input file to output file. *
3 ************************************************/
4 #include <iostream>
5 #include <unistd.h>
6 #include <fcntl.h>
7
8 int main() {
9         // The fd of the input file
10        int in_fd = open("file.in", O_RDONLY);
11
12        // The fd of the output file
13        int out_fd = open("file.out",
14                                O_WRONLY|O_CREAT, 0666);
15
16        char ch; // Character to copy
17
18        if (in_fd < 0) {
19                std::cout <<
20                "Error could not open input file\n";
21                exit (8);
22        }
23
24        if (out_fd < 0) {
25                std::cout <<
26                        "Error could not open output file\n";
27                exit (8);
28        }
29        while (1) {
30                if (read(in_fd, &ch, 1) != 1)
31        break;
32
33        write(out_fd, &ch, 1);
34        }
35        close(in_fd);
36        close(out_fd);
37        return (0);
38 }
```

(راهنمایی ۶، جواب ۹۶)

[1] Slow but Sure

فصل ۳: کاراکتر عجیب[1]

برنامه های این فصل همه کار می کنند و آنچه را که قرار است انجام دهند، انجام می دهند بجز اینکه یک یا دو کاراکتر در جای خود قرار ندارند. مسلما این کاراکترها، به شگفتی های واقعی و خرابی های کلی منجر خواهند شد.

برنامه ۱۷: دوباره سلام[2]

بازهم این کار را انجام می دهیم. "Hello World" را خراب کردیم. مشکل چیست؟

```
1 #include <iostream>
2
3 int main()
4 {
5         std::cout << "Hello World!/n";
6         return (0);
7 }
```

(راهنمایی ۱۷۲، جواب ۶۹)

برنامه نویسان واقعی به زبان کوبول برنامه نمی نویسند. کوبول برای برنامه نویسان بی مایه مناسب است.

برنامه های برنامه نویسان واقعی هیچ وقت در اولین بار کار نمی کند. ولی اگر آنها را روی ماشین قرار دهید، می توانند کار کنند البته بعد از "تعداد بسیار کمی" جلسه ۳۰ ساعته برای دیباگ کردن.

برنامه نویسان واقعی هیچ وقت از ۹ تا ۵ کار نمی کنند. اگر برنامه نویس واقعی را حول و حوش ۹ صبح دیدید، بخاطر این است که تمام شب بیدار بوده اند.

برنامه نویسان واقعی هیچ وقت مستندسازی نمی کنند. مستندسازی برای ابلهانی است که نمی توانند کد برنامه را بخوانند.

برنامه نویسان واقعی به پاسکال، BLISS یا Ada یا هر کدام از آن زبانهای ریشه ای علوم کامپیوتری برنامه نمی نویسند. انواع داده قوی فقط بدرد افراد کم حافظه می خورد.

[1] One Character Wonders
[2] Hello Again

برنامه ۱۸: کلاسیک[1]

اگر شما یک برنامه نویس هستید، اشتباه موجود در برنامه زیر را مرتکب شده اید. اگر دارید یک برنامه نویس می شوید، این اشتباه را مرتکب خواهید شد. و این شما را سرگشته می کند تا موقعی که بفهمید قضیه چه بوده است. بنا براین برنامه زیر چه کار می کند؟

```
1 /************************************************
2 * Test the logic for a simple accounting *
3 * program. *
4 ************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         // Amount owed (if any) by the user
10        int amount;
11
12        std::cout << "Enter current balance: ";
13        std::cin >> amount;
14
15        if (amount = 0)
16                std::cout << "You owe nothing\n";
17        else
18                std::cout << "You owe " << amount << "\n";
19
20        return (0);
21 }
```

(راهنمایی ۱۵۵، جواب ۴۷)

من برای یک شرکت نرم افزاری مهم و روی یک نسخه بین المللی واژه پرداز کار می کردم. صفحه آغازین، تاریخ واگذاری را بصورت mm/dd/yy نشان می داد مثلا 09/20/83. ولی در اروپا شکل استاندارد تاریخ بصورت dd/mm/yy است. بعنوان راهنمایی از رئیس خود پرسیدم که از کدام شکل استفاده کنم. او این موضوع را به بحث گذاشت و حدود یک ماه با مدیران خود در مورد مسأله تبادل نظر کرد. او تا یک هفته بعد از اینکه من نرم افزار را تحویل دادم، به من جواب نداد. در این مدت من مسأله را با تنظیم تاریخ روی ۱۱ نوامبر حل کردم. بله تاریخ ما بدین صورت بود: 11/11/83.

[1] Classic

برنامه ۱۹: متهم ردیف اول[1]

این یک برنامه ساده است که اعداد اول بین ۲ تا ۹ را پیدا می کند. الگوریتم استفاده شده بسیار ساده است ولی با این حال انتظار می رود که درست کار کند، پس چه چیزی دارد واقعا اتفاق می افتد؟

```
1 /*************************************************
2 * prime -- A very dump program to check to see *
3 * if the numbers 2-9 are prime. *
4 *************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         int i; // Number we are checking
10
11        for (i = 2; i < 10; ++i) {
12                switch(i) {
13                        case 2:
14                        case 3:
15                        case 5:
16                        case 7:
17                                std::cout << i << " is prime\n";
18                                break;
19                        default:
20                                std::cout << i <<
21                                        " is not prime\n";
22                        break;
23                }
24        }
25        return (0);
26 }
```

(راهنمایی ۳۵۴، جواب ۶۷)

کامپیوتر مؤسسه رفاه اجتماعی در واشنگتن، سن افراد را بصورت دو رقمی ذخیره می کرد. سن بانویی برای سیستم خیلی زیاد بود. وقتی او ۱۰۰ سالش شد، کامپیوتر سن او را بصورت 00 ذخیره کرد و ۱۰۱ بصورت 01 ذخیره شد. این امر زیاد مشکل ساز نبود تا اینکه او به سن ۱۰۷ سالگی رسید و دولت یک مأمور آموزش و پرورش به خانه او فرستاد تا بررسی کند که چرا او در کلاس اول ثبت نام نکرده است.

[1] Prime Suspect

برنامه ۲۰: ساده تر از آنچه که انتظار می رود[1]

برنامه زیر قرار است یک لیست از مجذور اعداد ۱ تا ۱۰ تولید کند. یک لیست از مجذورها تولید می کند ولی آن چیزی نیست که برنامه نویس انتظار داشته است.

```
1 /*************************************************
2 * Print out the square of the numbers *
3 * from 1 to 10 *
4 *************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         int index; /* Index into the table */
10
11        for (index = 1; index <= 10; ++index);
12                std::cout << index << " squared " <<
13                        (index * index) << '\n';
14
15        return (0);
16 }
```

(راهنمایی ۱۹۳، جواب ۳۴)

برنامه نویسان واقعی به PL/I برنامه نمی نویسند. PL/I برای برنامه نویسانی است که نمی دانند به کوبول برنامه بنویسند یا به فرترن.

برنامه نویسان واقعی وقتی Adventure یا Rogue بازی می کنند، بهتر فکر می کنند.

برنامه نویسان واقعی به فرترن برنامه نمی نویسند. فرترن برای فرکانس فشار لوله و محاسبات کریستالوگرافی است. فرترن برای مهندسان ابلهی است که جورابهای تمیز و سفید می پوشند.

مهندسان نرم افزار واقعی، برنامه ها را دیباگ نمی کنند. آنها درستی برنامه را تشخیص می دهند. این فرایند مستلزم اجرای چیزی روی کامپیوتر نیست بجز احیانا یک بسته کمکی برای تشخیص درستی.

مهندسان نرم افزار واقعی ایده یک سخت افزار پیچیده و ثقیل در مسافتی دور که می تواند هر لحظه از کار بیفتد را دوست ندارند. آنها به افراد سخت افزاری بسیار بی اعتماد هستند و امیدوارند چنین سیستم هایی بصورت مجازی در هر سطحی وجود داشته باشد. آنها به کامپیوترهای شخصی علاقه دارند مگر اینکه بخواهند بسته نرم افزاری کمکی برای تشخیص درستی را اجرا کنند.

[1] Simpler Than Expected

برنامه ۲۱: بدون شرح[1]

برنامه زیر چه چیزی را چاپ می کند؟ چرا؟

```
1 /************************************************
2 * demonstrate how to do a divide. *
3 ************************************************/
4 #include <iostream>
5
6 /************************************************
7 * div -- Do a divide *
8 * *
9 * Returns: Result of the divide. *
10 * *
11 * divisor is reset to 1. *
12 ************************************************/
13 static int div(
14         int *divisor // Pointer to the divisor
15 )
16 {
17         int result = 5; // Dividend
18
19         result=result/*divisor; /* Do divide */;
20         *divisor=l;
21         return (result);
22 }
23
24 int main()
25 {
26         int num = 5; // Divisor
27
28         std::cout << "Division " <<
29                 div(&num) << std::endl;
30         return (0);
31 }
```

(راهنمایی ۱۶۸، جواب ۹۱)

جایزه بهترین خطای رمزآلود تقدیم می شود به:
Error: Success
من هنوز دارم رمز آنرا کشف می کنم.

[1] No Comment

برنامه ۲۲: برای پارامترهای ما زیاد است[1]

ایده برنامه زیر ساده است: با محدود کردن اندازه به **MAX**، مطمئن شوید که زیاد بزرگ نمی شود. کاری که می کنیم این است:

```
1 /*************************************************
2 * Test the logic to limit the size of a *
3 * variable. *
4 *************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         int size = 20; // Size to be limited
10        const int MAX = 25; // The limit
11
12        if (size > MAX)
13                std::cout << "Size is too large\n";
14                size = MAX;
15
16        std::cout << "Size is " << size << '\n';
17        return(0);
18 }
```

(راهنمایی ۳۰۴، جواب ۴)

دستور **true** در یونیکس کاری نمی کند. در واقع اولین نسخه این برنامه یک فایل دسته ای 0 خطی بود (به اصطلاح یونیکس shell script). در طول سالیان، اضافاتی بی معنی به آن افزوده شد تا جائیکه برنامه 0 خطی به صورت زیر درآمد.

```
#! /bin/sh
#
#       @(#)true.sh 1.5 88/02/07 SMI; from UCB
#
exit 0
```

عدد 1.5 شماره نسخه است. آن بدین معنی است که آنها چهار نسخه قبلی از این برنامه را دستکاری کردند تا به این نسخه رسیدند. دلیل اینکه چرا آنها یک برنامه پوچ را چهار بار تغییر دادند برای من قابل فهم نیست.

[1] Getting Too Big for Our Parameters

برنامه ۲۳: آنچه که گفتنی است[1]

برنامه نویس می خواهد نسخه **strlen** خودش را تست کند. تابع به اندازه کافی ساده است ولی شاید بیش از حد ساده است. بنابراین طول رشته های زیر چقدر است؟

Sam
This is a test
Hello World

```
1 /************************************************
2 * Compute the length of a string entered by *
3 * the user. *
4 ************************************************/
5 #include <iostream>
6
7 /************************************************
8 * length -- Find the length of a string *
9 * (strlen does a better job.) *
10 * *
11 * Returns: *
12 * length of the string. *
13 ************************************************/
14 static int length(
15         const char string[] // String to check
16 )
17 {
18         int index; // index into the string
19
20         /*
21         * Loop until we reach the
22         * end of string character
23         */
24         for (index=0; string[index] != '\0';++index)
25                 /* do nothing */
26
27         return (index);
28 }
29
30 int main()
31 {
32         char line[100]; // Input line from user
33
34         while (1) {
35                 std::cout << "Enter a string: ";
36                 std::cin.getline(line, sizeof(line));
37
38                 std::cout << "Length is " <<
39                         length(line) << '\n';
40         }
41         return (0);
42 }
```

(راهنمایی ۱۱۴، جواب ۹۷)

[1] The Long and the Short of It

برنامه ۲۴: بیشتر از یک تقسیم ساده[1]

این برنامه دو عدد صحیح را بر هم تقسیم می کند. با اینکه آنقدر ساده است که اشتباه نکند ولی اشتباه می کند.

```
1 /*************************************************
2 * Simple divide program. *
3 *************************************************/
4 #include <iostream>
5
6 int main()
7 {
8         int n1, n2; // Two integers
9
10        std::cout << "Enter two integers: ";
11        std::cin >> n1 >> n2;
12
13        if (n2 =! 0)
14                std::cout << "Result is: " <<
15                        (n1/n2) << '\n';
16        else
17                std::cout << "Can not divide by zero\n";
18
19        return (0);
20 }
```

(راهنمایی ۷۰، جواب ۲۵)

کاربران واقعی از این می ترسند که دستگاه را داغان کنند، ولی هیچ وقت از اینکه صورت شما را داغان کنند نمی ترسند.

کاربران واقعی ترکیب عجیب و غریبی از مقادیر ورودی بدست می آورند که سیستم را برای روزها از کار می اندازد.

کاربران واقعی از برنامه نویسان واقعی متنفرند.

برنامه نویسان واقعی از کاربران واقعی نفرت ندارند. برنامه نویسان واقعی کاربران واقعی را فقط آدم های نامربوط می دانند.

کاربران واقعی شماره تلفن منزل شما را می دانند.

کاربران واقعی هیچ وقت نمی دانند چه می خواهند، ولی همیشه می دانند که چه وقت برنامه شما آن کار را انجام نمی دهد.

کاربران واقعی هیچ وقت از کلید Help استفاده نمی کنند.

[1] Overly Simple Division

برنامه ۲۵: بیشترین شگفتی[1]

حلقه برنامه زیر برای چاپ یک پیغام خوش آمدگویی به اندازه ۱۰ بار طراحی شده است. ولی برنامه کار دیگری انجام می دهد. قضیه چیست؟

توجه: این برنامه روی کامپایلرهای GNU و دیگر سیستم هایی که رهنمون های پیش پردازنده را پیاده سازی نمی کنند، کامپایل نمی شود.

```
1 /************************************************
2 * Print a bunch of greetings. *
3 ************************************************/
4 #include <iostream>
5
6 #define MAX =10
7
8 int main()
9 {
10        int counter; // Current greeting
11
12        for (counter =MAX; counter > 0; --counter)
13                std::cout <<"Hi there\n";
14
15        return (0);
16 }
```

(راهنمایی ۱۹۴، جواب ۱۱۲)

مرکز کامپیوتر یک دانشگاه بزرگ در یک ساختمان قدیمی قرار داشت. آنها تقریبا یک مشکل آزاردهنده داشتند. شب هنگام وقتی که اپراتور، اتاق را ترک می کرد، کامپیوتر reboot می شد.

یک تکنیسین کامپیوتر فراخوانده می شد و به سرعت دریافت که سیستم فقط وقتی reboot می شود که اپراتور به حمام می رود. وقتی می رفت آب بخورد هیچ اتفاقی نمی افتاد.

یک سری از تکنیسین ها فراخوانده شدند تا مسأله را بررسی کنند. تجهیزات تشخیصی بسیاری روی کامپیوتر قرار گرفتند.

نهایتا ریشه مشکل را پیدا کردند. زمین آن ساختمان به لوله های آب وصل بود. وزن اپراتور حدود ۳۰۰ پوند بود و وقتی روی حمام می نشست آنرا قدری خم می کرد و لوله ها جدا می شدند. این امر اتصال با زمین را قطع می کرد و باعث یک نوسان کوچک می شده که کامپیوتر را reboot می کرد.

[1] Maximum Surprise

برنامه ۲۶: منطقه دردسر[1]

این برنامه قرار است تعیین کند که پهنا و درازا، زیاد کوچک نشوند. برای پهنا کار می کند ولی با درازا مشکل دارد.

```
1 /*************************************************
2 * Test the logic to limit the width and height *
3 * of a rectangle. *
4 *************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         // The smallest legal value
10        // of width and height
11        const int MIN = 10;
12
13        int width = 5; // Current width
14        int height = 50; // Current height
15
16        if (width < MIN) {
17                std::cout << "Width is too small\n";
18                width = MIN;
19
20        if (height < MIN)
21                std::cout << "Height is too small\n";
22                height = MIN;
23        }
24
25        std::cout << "area(" << width << ", " <<
26                height << ")=" <<
27                (width * height) << '\n';
28        return (0);
29 }
```

(راهنمایی ۲۹۰، جواب ۱۳)

[1] Trouble Area

فصل ۴: مسائل روزمره[1]

برنامه نویسان حرفه ای، برنامه های جدیدی می نویسند. این برنامه نویسان، هر روز مرتکب اشتباهاتی می شوند. اینها، اشتباهات ساده یک برنامه نویس مبتدی نیستند و آنقدر هم پیچیده نمی باشند که به عنوان مسائل پیشرفته در نظر گرفته شوند. این باگ ها، باگ های روزمره هستند.

برنامه ۲۷: "و" و "وو"[2]

این برنامه طراحی شده است تا بررسی کند که آیا دو عدد مخالف صفر می باشند یا نه. مشکل اینجاست که برنامه نویس زیاد از مختصر نویسی استفاده کرده است و یک جای کار ایراد دارد:

```
1 /************************************************
2 * if_test -- Simple test of the if statement. *
3 ************************************************/
4 #include <iostream>
5
6 int main()
7 {
8         int i1 = 12; // A number
9         int i2 = 3; // Another number
10
11        if (i1 & i2)
12                std::cout << "Both numbers are non-zero\n";
13        else
14                std::cout << "At least one number is zero\n";
15        return (0);
16 }
```

(راهنمایی ۳۵۱، جواب ۱۷)

یک منشی یادداشتی را فراهم کرده بود و نمی توانست آنرا ذخیره کند. "آیا فضای کافی برای ذخیره دارید؟"، سوالی بود که کارشناس کامپیوتر از او پرسید.

منشی جواب داد:"آه البته ، پیغامی دریافت می کنم که می گوید Disk space OK ".

کارشناس کامپیوتر از بالای سر او نگاه کرد و مطمئن شد که پیغام Disk space: OK واقعا وجود دارد.

سپس او چند فایل را پاک کرد و پیغام تبدیل شد به Disk space: 4K. بعد از پاک کردن چند فایل دیگر، پیغام تبدیل شد به Disk space: 32K و منشی می توانست یادداشت خود را ذخیره کند.

[1] Everyday Rroblems
[2] "and" and "and and"

برنامه ۲۸: خطای صفر[1]

برنامه زیر طراحی شده است تا یک آرایه را صفر کند. پس چرا کار نمی کند؟ آیا **memset** خراب شده است؟

```
1 /************************************************
2 * zero_array -- Demonstrate how to use memset *
3 * to zero an array. *
4 ************************************************/
5 #include <iostream>
6 #include <cstring>
7
8 int main()
9 {
10        // An array to zero
11        int array[5] = {1, 3, 5, 7, 9};
12
13        // Index into the array
14        int i;
15
16        // Zero the array
17        memset(array, sizeof(array), '\0');
18
19        // Print the array
20        for (i = 0; i < 5; ++i)
21        {
22                  std::cout << "array[" << i << "]= " <<
23                  array[i] << std::endl;
24        }
25        return (0);
26 }
```

(راهنمایی ۵۰، جواب ۲۰)

از یک راهنمای فرترن برای کامپیوترهای Xerox:

هدف اصلی عبارت **DATA** این است که به ثوابت، اسم اختصاص دهد. بجای اینکه هر بار بجای π بنویسیم 3.141592653589793، می توانیم با عبارت **DATA**، این مقدار را به متغیر PI اختصاص دهیم و بجای آن عدد طولانی از آن استفاده کنیم. این کار همچنین تغییر دادن برنامه را آسانتر می کند چون ممکن است مقدار π عوض شود.

[1] Zero Error

برنامه ۲۹: خواننده عزیز، این خیلی ابتدایی است[1]

برنامه زیر طراحی شده است تا یک ماتریس ۳ در ۳ را نمایش دهد. ولی نتایج، عناصر ماتریس نیستند؛ آنها چیزهای دیگری هستند. قضیه چیست؟

```
1 /************************************************
2 * print_element -- Print an element in a *
3 * matrix. *
4 ************************************************/
5 #include <iostream>
6
7 // A simple matrix
8 int matrix[3][3] = {
9         {11, 12, 13},
10        {21, 22, 23},
11        {31, 32, 33}
12 };
13
14 int main()
15 {
16        std::cout << "Element[1,2] is " <<
17                  matrix[1,2] << std::endl;
18        return (0);
19 }
```

(راهنمایی ۸۹، جواب ۸۶)

یک برنامه نقشه کشی را می شناسم که متملقانه ترین پیغام خطایی را تاکنون وجود داشته نمایش می دهد:

این برنامه حقیر و بی ارزش مفتخر است به اطلاع حضرتعالی برساند که نمی توانم مقدار مقیاس ۱۰۰۰ را از شما بپذیرم زیرا برنامه نویس بی ملاحظه و کوته فکری که مرا نوشته است، مقدار این متغیر را بین ۱ و ۱۰۰ محدود ساخته است.

[1] It's Elementary, My Dear Reader

برنامه ۳۰: کمی دردسر[1]

این برنامه از یک متغیر برای نگهداری هشت علامت اجازه استفاده می کند. برنامه نویس می خواهد که اجازه های مدیریت (P_ADMIN) و پشتیبانی داده ها (P_BACKUP) را به کاربر خاصی بدهد و بعد بررسی کند که آیا بیت ها درست مقداردهی شده اند یا نه. چه اتفاقی دارد می افتد؟

```
1 /************************************************
2 * print_privs -- Print some of the privilege *
3 * flags. *
4 ************************************************/
5 #include <iostream>
6
7 #define CI const int
8 CI P_USER = (1 << 1); // Normal user privileges
9 CI P_REBOOT = (1 << 2); // Can reboot systems
10 CI P_KILL = (1 << 3); // Can kill any process
11 CI P_TAPE = (1 << 4); // Can use tape devices
12 CI P_RAW = (1 << 5); // Can do raw io
13 CI P_DRIVER = (1 << 6); // Can load drivers
14 CI P_ADMIN = (1 << 7); // Can do administration
15 CI P_BACKUP = (1 << 8); // Can do backups
16
17 int main()
18 {
19         // The privileges
20         unsigned char privs = 0;
21
22         // Set some privs
23         privs |= P_ADMIN;
24         privs |= P_BACKUP;
25
26         std::cout << "Privileges: ";
27
28         if ((privs & P_ADMIN) != 0)
29                 std::cout << "Administration ";
30
31         if ((privs & P_BACKUP) != 0)
32                 std::cout << "Backup ";
33
34         std::cout << std::endl;
35         return (0);
36 }
```

(راهنمایی ۷، جواب ۱۱)

[1] A Bit of Trouble

برنامه ۳۱: اعداد بسیار کوچک[1]

این برنامه زیرک بود. می خواست از بیتها برای ذخیره سازی علامتها استفاده کند تا از مشکلی که در برنامه ۳۰ بوجود آمد جلوگیری کند. ولی او هم مشکلات جدیدی ایجاد کرد:

```
1 /*************************************************
2 * printer status -- Print the status of the *
3 * printer. *
4 *************************************************/
5 #include <iostream>
6
7 /*
8 * Printer status information.
9 */
10 struct status {
11         // True if the printer is on-line
12         int on_line:1;
13
14         // Is the printer ready
15         int ready:1;
16
17         // Got paper
18         int paper_out:1;
19
20         // Waiting for manual feed paper
21         int manual_feed:1;
22 };
23
24 int main()
25 {
26         // Current printer status
27         status printer_status;
28
29         // Tell the world we're on-line
30         printer_status.on_line = 1;
31
32         // Are we on-line?
33         if (printer_status.on_line == 1)
34                 std::cout << "Printer is on-line\n";
35         else
36                 std::cout << "Printer down\n";
37         return (0);
38 }
```

(راهنمایی ۱۶۷، جواب ۴۲)

[1] Very Small Numbers

برنامه ۳۲: مشکل دو چندان[1]

چرا نمی توانیم هیچ وقت کاراکترهای دوتایی را پیدا کنیم؟

```
1 /************************************************
2 * test the find_double array. *
3 ************************************* **********/
4 #include <iostream>
5 char test[] = "This is a test for double letters\n";
6 /************************************************
7 * find_double -- Find double letters in an *
8 * array. *
9 * *
10 * Returns: *
11 * number of double letters in a string. *
12 ************************************************/
13 static int find_double(
14         const char str[] // String to check
15 ) {
16         int index; // Index into the string
17
18         for (index = 0; str[index] != '\0'; ++index) {
19                 /*
20                 * Start prev_ch out with a strange value
21                 * so we don't match on the first
22                 * character of the string.
23                 */
24                 char prev_ch = '\0';
25
26                 if (prev_ch == str[index])
27                         return (index-1);
29                 prev_ch = str[index];
30         }
31         return (-1);
32 }
33
34 int main() {
35         std::cout << "find_double= " <<
36                 find_double(test) << std::endl;
37         return (0);
38 }
```

(راهنمایی ۲۶۱، جواب ۱۰۶)

[1] Double Trouble

برنامه ۳۳: پست فطرت ها[1]

برنامه زیر باید ABC را بعنوان خروجی بدهد. ولی واقعا چه کار می کند؟

```
1 /************************************************
2 * Toy program to print three characters. *
3 ************************************************/
4 #include <iostream>
5
6 int main()
7 {
8         //A character to be printed
9         char ch = 'A';
10
11        std::cout << ch; // Output A
12        std::cout << ch+1; // Output B
13        std::cout << ch+2; // Output C
14        std::cout << std::endl;
15        return (0);
16 }
```

(راهنمایی ۱۲۴، جواب ۴۵)

قانون کمترین شگفتی
برنامه باید طوری رفتار کند که به کمترین میزان، کاربر را شگفت زده سازد.

[1] Bad Characters

برنامه ۳۴: سنت ها چه شدند؟[1]

این یک برنامه ساده دفترچه چک است. برنامه برای مدتی بدرستی کارمی کند ولی بعد از اینکه تعداد زیادی ورودی اضافه شدند، مجموع کل چند سنت کم دارد. چه بر سر پولها آمده است؟

```
1 /*************************************************
2 * check -- Very simple checkbook program. *
3 * *
4 * Allows you to add entries to your checkbook *
5 * and displays the total each time. *
6 * *
7 * Restrictions: Will never replace Quicken. *
8 *************************************************/
9 #include <iostream>
10 #include <fstream>
11 #include <string>
12 #include <vector>
13 #include <fstream>
14 #include <iomanip>
15
16 /*************************************************
17 * check_info -- Information about a single *
18 * check *
19 *************************************************/
20 class check_info {
21         public:
22                 // Date the check was written
23                 std::string date;
24
25                 // What the entry is about
26                 std::string what;
27
28                 // Amount of check or deposit
29                 float amount;
30         public:
31                 check_info():
32                 date(""),
33                 what(""),
34                 amount(0.00)
35                  {};
36                 // Destructor defaults
37                 // Copy constructor defaults
38                 // Assignment operator defaults
39         public:
40                 void read(std::istream &in file);
41                 void print(std::ostream &out_file);
42 };
43
44 // The STL vector to hold the check data
45 typedef std::vector<check_info> check_vector;
46
47 /*************************************************
48 * check_info::read -- Read the check *
49 * information from a file. *
50 * *
51 * Warning: Minimal error checking *
52 *************************************************/
53 void check_info::read(
54         std::istream &in_file // File for input
55 ) {
56         std::getline(in_file, date);
57         std::getline(in_file, what);
58         in_file >> amount;
59         in_file.ignore(); // Finish the line
60 }
```

[1] Non-Cents

```
61 /************************************************
62 * check_info::print -- Print the check *
63 * information to a report. *
64 ************************************************/
65 void check_info::print(
66         std::ostream &out_file // File for output
67 ) {
68         out_file <<
69                 std::setiosflags(std::ios::left) <<
70                 std::setw(10) << date <<
71                 std::setw(50) << what <<
72                 std::resetiosflags(std::ios::left) <<
73                 std::setw(8) << std::setprecision(2) <<
74                 std::setiosflags(std::ios::fixed) <<
75                 amount << std::endl;
76 }
77
78 int main()
79 {
80         // Checkbook to test
81         check_vector checkbook;
82
83         // File to read the check data from
84         std::ifstream in_file("checks.txt");
85
86         if (in_file.bad()) {
87                 std::cerr << "Error opening input file\n";
88                 exit (8);
89         }
90         while (1) {
91                 check_info next_info; // Current check
92
93                 next_info.read(in_file);
94                 if (in_file.fail())
95                         break;
96
97                 checkbook.push_back(next_info);
98         }
99         double total = 0.00; // Total in the bank
100        for (check_vector::iterator
101                cur_check = checkbook.begin();
102                cur_check != checkbook.end();
103                cur_check++)
104        {
105                cur_check->print(std::cout);
106                total += cur_check->amount;
107        }
108        std::cout << "Total " << std::setw(62) <<
109        std::setprecision(2) <<
110        total << std::endl;
111        return (0);
112 }
```

(راهنمایی ۳۹، جواب ۱۰۷)

برنامه ۳۵: حالا می خواهید یک میلیون را چاپ کنید[1]

من نمی دانستم که می توانیم در ثوابت C++، ویرگول داشته باشیم. خب چرا برنامه زیر کامپایل می شود؟ اصلا چه کار می کند؟

```
1 /*************************************************
2 * print the value on one million. *
3 *************************************************/
4 #include <iostream>
5
6 int main()
7 {
8         // Variable to hold a million
9         long int one_million;
10
11        // Set the variable
12        one_million = 1,000,000;
13
14        std::cout <<
15                "One million " << one_million <<
16                std::endl;
17        return (0);
18 }
```

(راهنمایی ۵۵، جواب ۴۴)

س: برای تعویض یک لامپ به چند برنامه نویس نیاز است؟

س: برای تعویض یک لامپ به چند برنامه نویس مایکروسافت نیاز است؟

جواب:
- هیچی، این یک مشکل سخت افزاری است.
- هیچی، مایکروسافت، تاریکی را بعنوان جدیدترین نوآوری درعرصه تکنولوژی ارائه کرده است.

[1] So You Want to Print a Million

برنامه ۳۶: انباشت بیش از حد[1]

چرا این برنامه، فضای پشته را پر می کند؟

```
1 /*************************************************
2 * test the data_holder class. *
3 *************************************************/
4 #include <iostream>
5 /*************************************************
6 * data_holder -- A class to hold a single *
7 * integer *
8 * *
9 * Member functions: *
10 * get -- Get value *
11 * *
12 * Note: By default the value of the data is 5. *
13 * *
14 * Warning: More member functions need to be *
15 * added to this to make it useful. *
16 *************************************************/
17 class data_holder {
18         private:
19                 int data; // Data to store
20         public:
21                 // Constructor -- Set value to default (5)
22                 data_holder(void):data(5) {};
23
24                 // Destructor defaults
25                 //
26                 // Copy constructor
27                 data_holder(const data_holder &old) {
28                         *this = old;
29                 }
30
31                 // Assignment operator
32                 data_holder operator = (
33                         data_holder old_data_holder) {
34                                 data = old_data_holder.data;
35                                 return (*this);
36                 }
37
38                 // Get the data item
39                 int get(void)
40                 {
41                         return (data);
42                 }
43 };
44
45 int main() {
46         // A data holder
47         data_holder var1;
48
49         // Copy of a data holder
50         data_holder var2(var1);
51         return (0);
52 }
```

(راهنمایی ۵۳، جواب ۱۲)

[1] Stacked Too High

برنامه ۳۷: این برنامه یک نکته دارد[1]

برنامه زیر طراحی شده است تا یک آرایه را صفر کند، ولی بعضی مواقع کارهای دیگری می کند.

```
1 /*************************************************
2 * Pointer demonstration. *
3 *************************************************/
4 #include <iostream>
5
6 static int data[16]; // Data to be stored
7 static int n_data = 0; // Number of items stored
8
9 int main()
10 {
11          int *data_ptr; // Pointer to current item
12
13          // Zero the data array
14          for (data_ptr = data+16-1;
15                    data_ptr >= data;
16                    --data_ptr)
17          {
18                    *data_ptr = 0;
19          }
20
21          // Enter data into the array
22          for (n_data = 0; n_data < 16; ++n_data) {
23                    std::cout <<
24                              "Enter an item or 0 to end: ";
25                    std::cin >> data[n_data];
26
27                    if (data[n_data] == 0)
28                              break;
29          }
30
31          // Index for summing
32          int index;
33
34          // Total of the items in the array
35          int total = 0;
36
37          // Add up the items in the array
38          for (index = 0; index < n_data; ++index)
39                    total += data[index];
40
41          // Print the total
42          std::cout << "The total is: " <<
43                    total << std::endl;
44
45          return (0);
46 }
```

(راهنمایی ۸۷، جواب ۲۱)

[1] This Problem Has a Point

برنامه ۳۸: مقدار درست[1]

این، قسمتی از یک برنامه واضح است. خب، چه چیزی را واقعا چاپ می کند؟

File: main.cpp

```
1 /************************************************
2 * test the check_for_even function. *
3 ************************************************/
4 #include <iostream>
5
6 int value = 21; // Value of the system size
7
8 // Checks global value for even or not.
9 extern void check_for_even(void);
10
11 int main(void)
12 {
13         check_for_even();
14         std::cout << "Value is " << value << '\n';
15         return (0);
16 }
```

File: check.cpp

```
1 #include <iostream>
2
3 // Value of the control system size
4 int value = 30;
5
6 /************************************************
7 * check_for_even -- Check to see if global *
8 * value is even. *
9 ************************************************/
10 void check_for_even(void)
11 {
12         if ((value % 2) == 0)
13                 std::cout << "Ok\n";
14         else
15                 std::cout << "Value problem\n";
16 }
```

(راهنمایی ۲۴۸، جواب ۵۷)

[1] Good Value

برنامه ۳۹: تجدید نظر در ریاضیات دبستانی[1]

همه ما می دانیم که ۲ = ۱ + ۱ و ۳ = ۱ + ۱ + ۱. همچنین ۱/۳ + ۱/۳ + ۱/۳ برابر ۳/۳ یا ۱ است. برنامه کامپیوتری زیر این مسأله را به تصویر می کشد. ولی به دلایلی درست کار نمی کند. چرا؟

```
1 /************************************************
2 * test out basic arithmetic that we learned in *
3 * first grade. *
4 ************************************************/
5 #include <iostream>
6
7 int main()
8 {
9         float third = 1.0 / 3.0; // The value 1/3
10        float one = 1.0; // The value 1
11
12        if ((third+third+third) == one)
13        {
14                std::cout <<
15                "Equal 1 = 1/3 + 1/3 + 1/3\n";
16        }
17        else
18        {
19                std::cout <<
20                "NOT EQUAL 1 != 1/3 + 1/3 + 1/3\n";
21        }
22        return (0);
23 }
```

(راهنمایی ۱۱۳، جواب ۵۴)

دانش آموزی اولین برنامه بیسیک خود را نوشته بود و با دستور RUN می خواست آنرا اجرا کند. کامپیوتر یک سری عدد نشان داد و به سرعت صفحه بالا می رفت و اعداد بیشتری نمایش داده می شد و دانش آموز بیچاره نمی توانست آنها را بخواند.

دانش آموز دقیقه ای فکر کرد و از خود پرسید: "آیا اگر می نوشتم WALK، آرامتر اجرا می شد؟".

[1] Kindergarten Arithmetic Revision

برنامه ۴۰: دقت غیرقابل باور[1]

این برنامه طراحی شده تا دقت اعداد اعشاری را تشخیص دهد. ایده آن ساده است. مقادیر زیر را محاسبه کن تا جاییکه اعداد با هم برابر شوند:

1.0 == 1.5 (1 + ½ or 1+ 1/21) (1.1 binary)
1.0 == 1.25 (1 + ¼ or 1 + 1/22) (1.01 binary)
1.0 == 1.125 (1 + 1/8 or 1+ 1/23) (1.001 binary)
1.0 == 1.0625 (1 + 1/16 or 1+ 1/24) (1.0001 binary)
1.0 == 1.03125 (1 + 1/32 or 1 + 1/25) (1.00001 binary)

این کار به ما دقت محاسبات را می دهد. این برنامه روی یک کامپیوتر PC و اعداد اعشاری ۳۲ بیت اجرا شد. خب، انتظار داریم چند رقم دودویی در یک عدد اعشاری ۳۲ بیتی باشد؟ این برنامه جواب درست را به ما نمی دهد. چرا؟

```
1 /************************************************
2 * accuracy test. *
3 * *
4 * This program figures out how many bits *
5 * accuracy you have on your system. It does *
6 * this by adding up checking the series: *
7 * *
8 * 1.0 == 1.1 (binary) *
9 * 1.0 == 1.01 (binary) *
10 * 1.0 == 1.001 (binary) *
11 * .... *
12 * *
13 * Until the numbers are equal. The result is *
14 * the number of bits that are stored in the *
15 * fraction part of the floating point number. *
16 ************************************************/
17 #include <iostream>
18
19 int main()
20 {
21         /* two numbers to work with */
22         float number1, number2;
23
24         /* loop counter and accuracy check */
25         int counter;
26
27         number1 = 1.0;
28         number2 = 1.0;
29         counter = 0;
30
31         while (number1 + number2 != number1) {
32                 ++counter; // One more bit accurate
33
34                 // Turn numbers like 0.1 binary
35                 // into 0.01 binary.
36                 number2 = number2 / 2.0;
37         }
38         std::cout << counter << " bits accuracy.\n";
39         return (0);
40 }
```

(راهنمایی ۳۵۲، جواب ۷۳)

[1] Unbelievable Accuracy

برنامه ۴۱: کمی دردسر[1]

bit_out یک مقدار ۱۶ بیتی را می گیرد و مقدار هر بیت را چاپ می کند. این، یک ارائه گرافیکی از مقدار درست می کند ولی خروجی کمی عجیب و غریب است. مشکل کجاست؟

```
1 /*************************************************
2 * bit test -- Test the routine to print out *
3 * the bits in a flag. *
4 *************************************************/
5 #include <iostream>
6 /*************************************************
7 * bit_out -- print a graphical *
8 * representation of each bit in a *
9 * 16 bit word. *
10 * *
11 * For example: *
12 * 0×55AF will print -X-X-X-XX-X-XXXX *
13 *************************************************/
14 void bit_out(
15         const short int value // Value to print
16 )
17 {
18         // The bit we are printing now
19         short int bit = (1<<16);
20
21         int count; // Loop counter
22
23         for (count = 0; count < 16; ++count)
24         {
25                 if ((bit & value) != 0)
26                         std::cout << "X";
27                 else
28                         std::cout << '-';
29                 bit >>= 1;
30         }
31         std::cout << std::endl;
32 }
33 int main()
34 {
35         bit_out(0×55AF);
36         return (0);
37 }
```

(راهنمایی ۳۳۲، جواب ۲)

[1] A Bit of Trouble

برنامه ۴۲: کمی دردسر بیشتر[1]

ما با تغییر خط ۱۹، برنامه ۴۱ را درست کردیم. خب پس حالا برنامه کار می کند، درست است؟ البته که نه. یک برنامه که درست کار می کند، جایش در این کتاب نیست.

```
1 /*************************************************
2 * bit test -- Test the routine to print out *
3 * the bits in a flag. *
4 *************************************************/
5 #include <iostream>
6 /*************************************************
7 * bit_out -- print a graphical *
8 * representation of each bit in a *
9 * 16 bit word. *
10 * *
11 * For example: *
12 * 0×55AF will print -X-X-X-XX-X-XXXX *
13 *************************************************/
14 void bit_out(
15         const short int value // Value to print
16 )
17 {
18         // The bit we are printing now
19         short int bit = (1<<15);
20
21         int count; // Loop counter
22
23         for (count = 0; count < 16; ++count)
24         {
25                 if ((bit & value) != 0)
26                         std::cout << "X";
27                 else
28                         std::cout << '-';
29                 bit >>= 1;
30         }
31         std::cout << std::endl;
32 }
33 int main()
34 {
35         bit_out(0×55AF);
36         return (0);
37 }
```

(راهنمایی ۱۸۰، جواب ۱۹)

[1] A Bit More Trouble

برنامه ۴۳: بی اساس[1]

می دانیم که 2 یک int است. پس چرا C++ فکر می کند که آن، یک عدد اعشاری است و تابع غلط را فراخوانی می کند؟

```
1 /*************************************************
2 * demonstrate the use of derived classes. *
3 *************************************************/
4 #include <iostream>
5
6 /*************************************************
7 * base -- A sample base class. *
8 * Prints various values. *
9 *************************************************/
10 class base
11 {
12         // Constructor defaults
13         // Destructor defaults
14         // Copy constructor defaults
15         // Assignment operator defaults
16         public:
17                 // Print a floating point number
18                 void print_it(
19                         float value // The value to print
20                 )
21                 {
22                         std::cout <<
23                                 "Base (float=" << value << ")\n";
24                 }
25                 // Print an integer value
26                 void print_it(
27                         int value // The value to print
28                 )
29                 {
30                         std::cout <<
31                                 "Base (int=" << value << ")\n";
32                 }
33 };
34
35 class der
36 {
37         // Constructor defaults
38         // Destructor defaults
39         // Copy constructor defaults
40         // Assignment operator defaults
41         public:
42                 // Print a floating point number
43                 void print_it(
44                         float value // The value to print
45                 )
46                 {
47                         std::cout <<
48                                 "Der (float=" << value << ")\n";
49                 }
50 };
51
52 int main()
53 {
```

[1] Baseless

```
54      der a_var; // A class to play with
55
56      // Print a value using der::print_it(float)
57      a_var.print_it(1.0);
58
59      // Print a value using base::print_it(int)
60      a_var.print_it(2);
61      return (0);
62 }
```

(راهنمایی ۳۳۰، جواب ۵۸)

نسخه اصلی دستور mt یونیکس یک پیغام خطای غیرعادی داشت که وقتی که نمی توانست دستوری را بفهمد، ظاهر می شد:

mt –f /dev/rst8 funny
mt: Can't grok funny

برای کسانی که با *غریبه در غربت* نوشته روبرت هاینلین آشنا نیستند،باید بگویم که *grok* معادل مریخی *فهمیدن* است.

این اصطلاح به کشورهای دیگر راه نیافت. یک برنامه نویس آلمانی با سادگی تمام رفت که معنی grok را در لغتنامه انگلیسی – آلمانی بیابد.

برنامه ۴۴: مسأله مرتب سازی[1]

برنامه زیر قرار است که تفاوت بین عناصر مجاور یک آرایه را بیابد. چرا درست کار نمی کند؟

```
1 /************************************************
2 * diff elements -- Print the differences *
3 * between adjacent elements of any array. *
4 ************************************************/
5 #include <iostream>
6
7 // Any array containing pairs of values.
8 // Ends with the sentinel -1
9 static int array[12] =
10 {
11         44, 8,
12         50, 33,
13         50, 32,
14         75, 39,
15         83, 33,
16         -1, -1
17 };
18
19 // Array to hold the differences
20 static int diff[6];
21
22 int main()
23 {
24         int i; // Index into the array
25
26         // Index into the diff results
27         int diff_index;
28
29         i = 0;
30         diff_index = 0;
31         // Difference adjacent elements of an array
32         while (array[i] != 0)
33         {
34                 diff[diff_index++] =
35                 array[i++] - array[i++];
36         }
37
38         // Print the results
39         for (i = 0; i < 6; ++i)
40         {
41                 std::cout << "diff[" << i << "]= " <<
42                 diff[i] << std::endl;
43         }
44         return (0);
45 }
```

(راهنمایی ۱۷۷، جواب ۲۶)

[1] Ordering Problem

برنامه ۴۵: شگفتی سه چندان[1]

آیا a, b, c به ترتیب نزولی هستند؟ آیا این برنامه با شما موافق است؟

```
1 /************************************************
2 * test to see if three variables are in order. *
3 ************************************************/
4 #include <iostream>
5
6 int main()
7 {
8         int a,b,c; // Three simple variables
9
10        a = 7;
11        b = 5;
12        c = 3;
13
14        // Test to see if they are in order
15        if (a > b > c)
16                std::cout << "a,b,c are in order\n";
17        else
18                std::cout << "a,b,c are mixed up\n";
19        return (o);
20 }
```

(راهنمایی ۳۱۲، جواب ۸۰)

دیباگر همه کامپیوترهای DEC، DDT نام دارد. در دفترچه راهنمای PDP-10 DDT پاورقی وجود دارد که این اسم از کجا آمده است:

پاورقی تاریخی: DDT در MIT و برای کامپیوتر PDP-1 در ۱۹۶۱ ابداع شد. در آن زمان DDT مخفف "DEC Debugging Tape" بود. از آن موقع، ایده یک برنامه دیباگ کننده آنلاین در کل صنایع کامپیوتر منتشر شد. برنامه های DDT اکنون برای همه کامپیوترهای DEC موجود می باشند. از آنجا که بجای نوار، امروزه از وسائل دیگری استفاده می شود، اسم مناسبتر "Dynamic Debugging Technique" بکار رفته است که مخفف آن همان DDT می شود. اختلاط معنی بین DDT-10 و یک آفت – کش معروف یعنی دیکلرو دیفنیل تریکلروتیل (C_{14} H_9 C_{15}) باید بسیار کم باشد چون هر کدام به یک نوع متفاوت و مانعه الجمع از آفت ها حمله می کنند.

[1] Triple Surprise

برنامه ۴۶: چیزی اشتباه نمی شود[1]

چرا این برنامه گاهی ....؟

```
1 /************************************************
2 * list -- Test out the command list decoder. *
3 * *
4 * Read a command from the input and check to *
5 * see if the command decoder can find it. *
6 ************************************************/
7 #include <iostream>
8 #include <cstring>
9
10 static inline void do_open() {
11         std::cout << "do_open called\n";
12 }
13 static inline void do_close() {
14         std::cout << "do_close called\n";
15 }
16 static inline void do_save() {
17         std::cout << "do_save called\n";
18 }
19 static inline void do_quit() {
20         exit(0);
21 }
22 /*
23 * The command as a string and
24 * as a function to execute
25 */
26 struct cmd_info {
27         char *const cmd;
28         void (*funct)();
29 };
30
31 /*
32 * List of all possible commands
33 */
34 static cmd_info cmd_list[] = {
35         {"open", do_open},
36         {"close", do_close},
37         {"save", do_save},
38         {"quit", do_quit},
39         {NULL, NULL}
40 };
41
42 /************************************************
43 * do_cmd -- Decode a command an execute it. *
44 * If the command is not found, output an *
45 * error. *
46 ************************************************/
47 static void do_cmd(
48         const char *const cmd
49 ) {
50         struct cmd_info *cur_cmd;
51
52         cur_cmd = cmd_list;
53
54         while (
```

[1] Nothing Goes Wrong

```
55                      (std::strcmp(cur_cmd->cmd, cmd) != 0) &&
56                              cur_cmd != NULL)
57          {
58                      cur_cmd++;
59          }
60          if (cur_cmd == NULL) {
61                      std::cout << "Command not found\n";
62          } else {
63                      cur_cmd->funct();
64          }
65 }
66
67 /***********************************************
68 * main -- Simple test program. *
69 ***********************************************/
70 int main()
71 {
72          char cmd[100];
73          while (1) {
74                      std::cout << "Cmd: ";
75                      std::cin.getline(cmd, sizeof(cmd));
76
77                      do_cmd(cmd);
78          }
79 }
```

(راهنمایی ۱۳۵، جواب ۷۰)

برنامه ۴۷: عقب ماندگی مایکروسافت[1]

چرا وقتی برنامه زیر در MS – DOS کامپایل و اجرا می شود نمی تواند فایل مربوطه را باز کند؟

```
/***********************************************
2 * read config file -- Open a configuration *
3 * file and read in the data. *
4 * *
5 * Designed to work on both UNIX and MS-DOS. *
6 * *
7 * Note: Incomplete program. *
8 ***********************************************/
9 #include <iostream>
10 #include <fstream>
11
12 #ifdef MS_DOS
13
14 // DOS path
15 const char name[] = "\root\new\table";
16
17 #else /* MS_DOS */
18
19 // UNIX path
20 const char name[] = "/root/new/table";
21
22 #endif /* MS_DOS */
23
24
25 int main() {
26         // The file to read
27         std::ifstream in_file(name);
28
29         if (in_file.bad())
30         {
31                 std::cerr <<
32                         "Error: Could not open " << std::endl;
33                 std::cerr << name << std::endl;
34                 exit (8);
35         }
36
37         return (0);
38 }
```

(راهنمایی ۲۱۷، جواب ۳۷)

[1] Microsoft Backwardness

برنامه ۴۸: نادانی های فایل[1]

برنامه زیر برای مدتی خوب کار می کند و سپس از پذیرش فایل هایی که عدد جادویی دارند سرباز می زند.

```
1 /*************************************************
2 * scan -- Scan a directory tree for files that *
3 * begin with a magic number. *
4 *************************************************/
5 #include <iostream>
6 #include <dirent.h>
7 #include <fcntl.h>
8 #include <unistd.h>
9
10 // Linux executable magic #
11 const long int MAGIC = 0×464c457f;
12
13 /*************************************************
14 * next_file -- find a list of files with *
15 * magic numbers that match the given *
16 * number. *
17 * *
18 * Returns the name of the file or *
19 * NULL if no more files. *
20 *************************************************/
21 char *next_file(
22         DIR *dir // Directory we are scanning
23 )
24 {
25         // The current directory entry
26         struct dirent *cur_ent;
27
28         while (1) {
29                 cur_ent = readdir(dir);
30                 if (cur_ent == NULL)
31                 return (NULL);
32
33                 // Open the fd for the input file
34                 int fd = open(cur_ent->d_name, 0_RDONLY);
35                 if (fd < 0)
36                         continue; // Can't get the file
37                 // so try again
38
39                 int magic; // The file's magic number
40
41                 // Size of the latest read
42                 int read_size =
43                         read(fd, &magic, sizeof(magic));
44
45                 if (read_size != sizeof(magic))
46                         continue;
47
48                 if (magic == MAGIC)
49                 {
50                         close(fd);
51                         return (cur_ent->d_name);
52                 }
53         }
54 }
```

[1] File Follies

```
55
56 /************************************************
57 * scan_dir -- Scan a directory for the *
58 * files we want. *
59 ************************************************/
60 void scan_dir(
61         const char dir_name[] // Directory name to use
62 )
63 {
64         // The directory we are reading
65         DIR *dir_info = opendir(dir_name);
66         if (dir_info == NULL)
67                 return;
68
69         chdir(dir_name);
70
71         while (1) {
72                 char *name = next_file(dir_info);
73                 if (name == NULL)
74                 break;
75                 std::cout << "Found: " << name << '\n';
76         }
77 }
78
79 int main()
80 {
81         scan_dir(".");
82         return (0);
83 }
```

(راهنمایی ۲۲۶، جواب ۶۰)

برنامه ۴۹: به سادگی گسستن یک ارتباط[1]

چرا برنامه زیر بعضی مواقع ... ؟

```
1 #include <iostream>
2 #include <string>
3 /************************************************
4 * linked_list -- Class to handle a linked list *
5 * containing a list of strings. *
6 * *
7 * Member functions: *
8 * add -- Add an item to the list *
9 * is_in -- Check to see if a string is *
10 * in the list. *
11 ************************************************/
12 class linked_list {
13         private:
14                 /*
15                 * Node in the list
16                 */
17                 struct node {
18                         // String in this node
19                         std::string data;
20
21                         // Pointer to next node
22                         struct node *next;
23                 };
24                 //First item in the list
25                 struct node *first;
26         public:
27                 // Constructor
28                 linked_list(void): first(NULL) {};
29                 // Destructor
30                 ~linked_list();
31         private:
32                 // No copy constructor
33                 linked_list(const linked_list &);
34
35                 // No assignment operator
36                 linked_list& operator = (const linked_list &);
37         public:
38                 // Add an item to the list
39                 void add(
40                         // Item to add
41                         const std::string &what
42                 ) {
43                         // Create a node to add
44                         struct node *new_ptr = new node;
45
46                         // Add the node
47                         new_ptr->next = first;
48                         new_ptr->data = what;
49                         first = new_ptr;
50                 }
51                 bool is_in(const std::string &what);
52 };
53 /************************************************
54 * is_in -- see if a string is in a *
```

[1] It's As Easy As Falling Off a Link

```
55 * linked list. *
56 * *
57 * Returns true if string's on the list, *
58 * otherwise false. *
59 ***********************************************/
60 bool linked_list::is_in(
61         // String to check for
62         const std::string &what
63 ) {
64         /* current structure we are looking at */
65         struct node *current_ptr;
66
67         current_ptr = first;
68
69         while (current_ptr != NULL) {
70                 if (current_ptr->data == what)
71                 return (true);
72
73                 current_ptr = current_ptr->next;
74         }
75         return (false);
76 }
77
78 /***********************************************
79 * linked_list::~linked_list -- Delete the *
80 * data in the linked list. *
81 ***********************************************/
82 linked_list::~linked_list(void) {
83         while (first != NULL) {
84                 delete first;
85                 first = first->next;
86         }
87 }
88
89 int main() {
90         linked_list list; // A list to play with
91
92         list.add("Sam");
93         list.add("Joe");
94         list.add("Mac");
95
96         if (list.is_in("Harry"))
97                 std::cout << "Harry is on the list\n";
98         else
99                 std::cout << "Could not find Harry\n";
100        return (0);
101 }
```

(راهنمایی ۱۸۶، جواب ۷۷)

یک بانوی نظافتچی، جای خراشی در کف اتاق کامپیوتر یافت و تصمیم گرفت که آنرا پاک کند. اول واکس را امتحان کرد، سپس پاک کننده آمونیاک و نهایتا بعنوان راه آخر از سیم ظرفشویی استفاده نمود. ترکیب این مواد، نه تنها برای جای خراش بلکه برای کامپیوترها هم کشنده بود.

روز بعد، وقتی اعضای اتاق، سر کار آمدند، تمام دستگاه هایشان از کار افتاده بود. با باز کردن دستگاه ها، متوجه شدند که اتصال کوتاه های بسیار زیادی در تمام مدارات رخ داده است.

چه اتفاقی افتاده بود؟ بانوی نظافتچی، ابتدائا لایه ای از واکس را به کف اتاق کشیده بود. آمونیاک، واکس را تبخیر کرده بود، که توسط هواکش های کامپیوترها به درون مکیده شده بود. لذا تمام مدارات توسط یک لایه از واکس

چسبناک پوشیده شده بود. تا اینجا زیاد بد نبود ولی بعد نوبت سیم ظرفشویی بود. براده های سیم ظرفشویی به درون دستگاه ها کشیده شده بود و به واکس موجود روی مدارات چسبیده بود و موجب اتصال کوتاه شده بود.

برنامه ۵۰: به هر حال، حقیقت چیست؟[1]

کامپیوترها عبارت "حقیقت شما را آزاد خواهد ساخت" را به "حقیقت شما را سرگشته خواهد ساخت" تبدیل کرده اند.

```
1 /*************************************************
2 * test bool_name, a function turn booleans into*
3 * text. *
4 *************************************************/
5 #include <iostream>
6 #include <string>
7
8 /*************************************************
9 * bool_name -- given a boolean value, return *
10 * the text version. *
11 * *
12 * Returns: *
13 * Strings "true" or "false" depending *
14 * on value. *
15 *************************************************/
16 static const std::string &bool_name(
17         const bool value // The value to check
18 )
19 {
20         // The "true" value
21         const std::string true_name("true");
22
23         // The "false" value
24         const std::string false_name("false");
25
26         if (value == true)
27                 return (true_name);
28
29         return (false_name);
30 }
31
32 int main() {
33         std::cout << "True is " <<
34         bool_name(true) << std::endl;
35
36         std::cout << "False is " <<
37         bool_name(false) << std::endl;
38         return (0);
39 }
```

(راهنمایی ۳۱۹، جواب ۳۰)

[1] What Is Truth, Anyway?

برنامه ۵۱: جمع زیادی[1]

برنامه نویس وقتی عملگرهای ++x, x++ را تعریف کرد، می خواست کار درست را انجام دهد. ولی با این حال برنامه زیر چه چیزی را چاپ می کند؟

```
1 /************************************************
2 * Demonstrate how to define and use increment *
3 * operator. *
4 ************************************************/
5 #include <iostream>
6
7 /************************************************
8 * num -- Class to hold a single number *
9 ************************************************/
10 class num
11 {
12         // Constructor defaults
13         // Destructor defaults
14         // Copy Constructor defaults
15         // Assignment operator defaults
16         public:
17                 // Value stored in the function
18                 int value;
19
20                 // Increment operator (i++)
21                 num operator ++(int)
22                 {
23                         num copy(*this); // Copy for return
24
25                         value++;
26                         return (copy);
27                 }
28
29                 // Increment operator (++i)
30                 num &operator ++(void)
31                 {
32                         value++;
33                         return (*this);
34                 }
35 };
36
37 int main()
38 {
39         num i; // A value to play with
40
41         i.value = 1;
42         ++++i;
43         std::cout << "i is " << i.value << std::endl;
44
45         i.value = 1;
46         i++++;
47         std::cout << "i is " << i.value << std::endl;
48         return (0);
49 }
```

(راهنمایی ۲۴۶، جواب ۸۷)

[1] A Surplus of Pluses

برنامه ۵۲: مورد مربع ناپدید شونده[1]

مساحت نمونه ما چیست؟

```
1 /*************************************************
2 * Demonstration of the rectangle class. *
3 *************************************************/
4 #include <iostream>
5
6 /*************************************************
7 * rectangle -- hold constant information about *
8 * a rectangle. *
9 * *
10 * Members: *
11 * area -- Area of the rectangle. *
12 * width -- width of the rectangle. *
13 * height - length of the rectangle. *
14 *************************************************/
15 class rectangle
16 {
17         public:
18                 const int area; // Rectangle's Area
19                 const int width; // Rectangle's Width
20                 const int height; // Rectangle's Height
21
22         public:
23                 // Create a rectangle and assign the
24                 // initial values
25                 rectangle(
26                         const int i_width, // Initial width
27                         const int i_height // Initial height
28                 ) : width(i_width),
29                         height(i_height),
30                         area(width*height)
31                 {}
32                 // Destructor defaults
33                 // Copy constructor defaults
34                 // Assignment operator defaults
35 };
36
37 int main()
38 {
39         // Rectangle to play with
40         rectangle sample(10, 5);
41
42         std::cout << "Area of sample is " <<
43                 sample.area << std::endl;
44         return (0);
45 }
```

(راهنمایی ۲۱۰، جواب ۹۳)

[1] The Case of the Disappearing Rectangle

برنامه ۵۳: سرگشتگی بیشینه[1]

تابع max ساده است، کد آزمون ساده است و جواب...، خب شما باید آنرا بیابید.

```
1 /************************************************
2 * test_max -- Test the max function. *
3 ************************************************/
4 #include <iostream>
5
6 /************************************************
7 * max -- return the larger of two integers. *
8 * *
9 * Returns: *
10 * biggest of the two numbers. *
11 ************************************************/
12 const int &max(
13         const int &i1, // A number
14         const int &i2 // Another number
15 )
16 {
17         if (i1 > i2)
18         return (i1);
19         return (i2);
20 }
21
22 int main()
23 {
24         // I is the biggest of the two expression
25         const int &i = max(1+2, 3+4);
26
27         std::cout <<
28                   "The biggest expression is " <<
29                   i << std::endl;
30
31         return (0);
32 }
```

(راهنمایی ۲۸۹، جواب ۲۲)

[1] Maximum Confusion

برنامه ۵۴: جهش به اعماق بیکران[1]

چرا این برنامه، حافظه کم می آورد؟

```
1 /************************************************
2 * Combine strings with a variable length *
3 * string class. *
4 ************************************************/
5 #include <setjmp.h>
6 #include <iostream>
7 #include <cstring>
8
9 // Place to store jump information
10 static jmp_buf top_level;
11
12 // Longest string combination allowed.
13 static const unsigned int MAX_LENGTH = 30;
14
15 /************************************************
16 * combine -- Combine two strings with *
17 * limit checking *
18 ************************************************/
19 static std::string combine(
20         const std::string &first, // First string
21         const std::string &second // Second string
22 )
23 {
24         // Strings put together
25         std::string together = first + second;
26
27         if (together.length() > MAX_LENGTH) {
28                 longjmp(top_level, 5);
29         }
30         return (together);
31 }
32
33 int main()
34 {
35         std::string first("First ");
36         int i;
37
38         for (i = 0; i < 10; i++) {
39
40                 // Save our place
41                 if (setjmp(top_level) == 0)
42                 {
43                         first = combine(first,
44                         std::string("second "));
45                 } else {
46                         std::cout <<
47                                 "Length limit exceeded\n";
48                         break;
49                 }
50         }
51         return (0);
52 }
```

(راهنمایی ۱۴۶، جواب ۶۶)

[1] Jumping off the Deep End

برنامه ۵۵: برنامه نویسی ابلهانه[1]

براون کشاورز، یک پرورش دهنده گوسفند، همسایه ای داشت که می توانست به یک گله نگاه کند و بگوید که چند گوسفند در آن وجود دارد. او بسیار متعجب بود که چگونه دوستش به این سرعت این کار را انجام می دهد، لذا از او پرسید.

"یان، چطور به این سرعت می توانی تعداد گوسفندها بگویی؟"

"به سادگی، پاها را می شمارم و بر ۴ تقسیم می کنم."

براون کشاورز آنقدر تحت تأثیر قرار گرفت که یک برنامه C++ کوتاه نوشت تا درستی الگوریتم گوسفندشماری یان را بررسی کند. این برنامه برای گله های بزرگ جواب نمی دهد. چرا؟

```
1 /************************************************
2 * sheep -- Count sheep by counting the *
3 * number of legs and dividing by 4. *
4 ************************************************/
5 #include <iostream>
6
7 /*
8 * The number of legs in some different
9 * size herds.
10 */
11 const short int small_herd = 100;
12 const short int medium_herd = 1000;
13 const short int large_herd = 10000;
14
15 /************************************************
16 * report_sheep -- Given the number of legs, *
17 * tell us how many sheep we have. *
18 ************************************************/
19 static void report_sheep(
20         const short int legs // Number of legs
21 )
22 {
23         std::cout <<
24                 "The number of sheep is: " <<
25                 (legs/4) << std::endl;
26 }
27
28 int main() {
29         report_sheep(small_herd*4); // Expect 100
30         report_sheep(medium_herd*4);// Expect 1000
31         report_sheep(large_herd*4); // Expect 10000
32         return (0);
33 }
```

(راهنمایی ۱۶۵، جواب ۱)

[1] Sheepish Programming

برنامه ۵۶: جادو از برنامه رخت بربسته است[1]

این برنامه طراحی شده است تا بررسی کند که آیا دو فایل در دو پوشه، دارای یک عدد جادویی می باشند یا نه.
در آزمون ما، فایلهای زیر موجود هستند:

first/first
second/second

هر دو این فایل ها، عدد جادویی را در خود دارند. این برنامه چه چیزی را و چرا نمایش می دهد؟

```
1 /*************************************************
2 * scan_dir -- Scan directories for magic files *
3 * and report the results. *
4 * *
5 * Test on the directories "first" and "second".*
6 *************************************************/
7 #include <iostream>
8 #include <dirent.h>
9 #include <fcntl.h>
10 #include <unistd.h>
11 const long int MAGIC = 0×464c457f; // Linux executable magic #
12 /*************************************************
13 * next_file -- find a list of files with magic *
14 * numbers that match the given number. *
15 * *
16 * Returns the name of the file or *
17 * NULL if no more files. *
18 *************************************************/
19 char *next_file(
20         DIR *dir // Directory to scan
21 ) {
22         // Current entry in the dir
23         struct dirent *cur_ent;
24
25         while (1) {
26
27                 cur_ent = readdir(dir);
28                 if (cur_ent == NULL)
29                 return (NULL);
30
31                 int fd = open(cur_ent->d_name, 0_RDONLY);
32                 if (fd < 0) {
33                         // Can't get the file so try again
34                         continue;
35                 }
36
37                 int magic; // The file's magic number
38
39                 // Size of the header read
40                 int read_size =
41                         read(fd, &magic, sizeof(magic));
42
43                 if (read_size != sizeof(magic)) {
44                         close(fd);
45                         continue;
46                 }
47
48                 if (magic == MAGIC) {
```

[1] The Magic is Gone from the Program

```
49                              close(fd);
50                              return (cur_ent->d_name);
51                      }
52                      close(fd);
53              }
54 }
55 /************************************************
56 * scan_dir -- Scan a directory for the files *
57 * we want. *
58 ************************************************/
59 char *scan_dir(
60              const char dir_name[] // Directory name to use
61 ) {
62              // Directory to scan
63              DIR *dir_info = opendir(dir_name);
64              if (dir_info == NULL)
65              return (NULL);
66
67              chdir(dir_name);
68
69              // Name of the file we just found
70              char *name = next_file(dir_info);
71              closedir(dir_info);
72
73              chdir(".."); // Undo the original chdir
74
75              return (name);
76 }
77
78 int main() {
79              // Find a file in the directory "first"
80              char *first_ptr = scan_dir("first");
81
82              // Find a file in the directory "second"
83              char *second_ptr = scan_dir("second");
84
85              // Print the information about the dir first
86              if (first_ptr == NULL) {
87                      std::cout << "First: NULL ";
88              } else {
89                      std::cout << "First: " << first_ptr << " ";
90              }
91              std::cout << '\n';
92
93              // Print the information about the dir second
94              if (second_ptr == NULL) {
95                      std::cout << "Second: NULL ";
96              } else {
97                      std::cout << "Second: " << second_ptr << " ";
98              }
99              std::cout << '\n';
100             return (0);
101 }
```

(راهنمایی ۸۶، جواب ۱۰۰)

مهندسان واقعی نرم افزار از ۹ تا ۵ کار می کنند، چون شغل آنها بطور رسمی اینگونه تعریف شده است. کار کردن تا دیروقت مانند استفاده از یک رویه خارجی مستند نشده[1] است.

[1] Undocumented external procedure

برنامه ۵۷: چگونه یک فایل را نخوانیم[1]

چه مشکلی از لحاظ قابلیت حمل در برنامه زیر وجود دارد؟

```
1 #include <iostream>
2
3 /*
4 * A data structure consisting of a flag
5 * which indicates which long int parameter
6 * follows.
7 */
8 struct data
9 {
10        // Flag indicating what's to follow
11        char flag;
12
13        // Value of the parameter
14        long int value;
15 };
16
17 /************************************************
18 * read_data -- Read data from the given file *
19 ************************************************/
20 void read_data(
21        std::istream &in_file, // File to read
22        struct data &what // Data to get
23 )
24 {
25        in_file.read(
26                dynamic_cast<char *>(&what),
27                sizeof(what));
28 }
```

(راهنمایی ۱۶۱، جواب ۷۱)

[1] How Not to Read a File

برنامه ۵۸: اسامی خارق العاده[1]

زیربزنامه tmp_name طراحی شده است تا اسم یک فایل موقتی را برگرداند. ایده این است که در هر بار فراخوانی، یک اسم منحصربفرد تولید شود: /var/tmp/tmp.0, /var/tmp/tmp.1, /var/tmp/tmp.2, …

اسامی تولید شده، یقینا منحصر بفرد هستند ولی نه آنگونه که برنامه نویس می خواست.

```
1 /************************************************
2 * tmp_test -- test out the tmp_name function. *
3 ************************************************/
4 #include <iostream>
5 #include <cstdio>
6 #include <cstring>
7 #include <sys/param.h>
8 /************************************************
9 * tmp_name -- return a temporary file name *
10 * *
11 * Each time this function is called, a new *
12 * name will be returned. *
13 * *
14 * Returns: Pointer to the new file name. *
15 ************************************************/
16 char *tmp_name(void) {
17         // The name we are generating
18         char name[MAXPATHLEN];
19
20         // The base of the generated name
21         const char DIR[] = "/var/tmp/tmp";
22
23         // Sequence number for last digit
24         static int sequence = 0;
25
26         ++sequence; /* Move to the next file name */
27
28         sprintf(name, "%s.%d", DIR, sequence);
29         return(name);
30 }
31 int main() {
32         char *a_name = tmp_name(); // A tmp name
33         std::cout << "Name: " << a_name << std::endl;
34         return(o);
35 }
```

(راهنمایی ۱۷۶، جواب ۱۸)

[1] Weird Name

برنامه ۵۹: فرزند اسامی خارق العاده[1]

برنامه زیر طراحی شده است تا با هر بار فراخوانی tmp_name یک اسم منحصر بفرد تولید شود. برای آزمایش آن، تصمیم گرفتیم که دو اسم را چاپ کنیم. هنوز هم برنامه کار نمی کند. چرا؟

```
1 /*************************************************
2 * test the tmp_name function. *
3 *************************************************/
4 #include <iostream>
5 #include <cstdio>
6 #include <cstring>
7 #include <sys/param.h>
8 /*************************************************
9 * tmp_name -- return a temporary file name. *
10 * *
11 * Each time this function is called, a new *
12 * name will be returned. *
13 * *
14 * Returns *
15 * Pointer to the new file name. *
16 *************************************************/
17 char *tmp_name(void)
18 {
19         // The name we are generating
20         static char name[MAXPATHLEN];
21
22         // The directory to put the temporary file in
23         const char DIR[] = "/var/tmp/tmp";
24
25         // Sequence number for last digit
26         static int sequence = 0;
27
28         ++sequence; /* Move to the next file name */
29
30         std::sprintf(name, "%s.%d", DIR, sequence);
31         return(name);
32 }
33
34 int main()
35 {
36         // The first temporary name
37         char *a_name = tmp_name();
38
39         // The second temporary name
40         char *b_name = tmp_name();
41
42         std::cout << "Name (a): " << a_name << endl;
43         std::cout << "Name (b): " << b_name << endl;
44         return(0);
45 }
```

(راهنمایی ۳۲۲، جواب ۶۴)

[1] Son of Weird Names

برنامه ۶۰: نوه اسامی خارق العاده[1]

خب، ما دوباره برنامه خود را درست کرده ایم و حالا از رشته های C++ استفاده می کنیم. ولی باز هم درست کار نمی کند. چرا؟

```
1 #include <iostream>
2 #include <string>
3
4 /************************************************
5 * tmp_name -- return a temporary file name *
6 * *
7 * Each time this function is called, a new *
8 * name will be returned. *
9 * *
10 * Returns *
11 * String containing the name. *
12 ************************************************/
13 std::string &tmp_name()
14 {
15         // The name we are generating
16         std::string name;
17
18         // Sequence number for last digit
19         static int sequence = 0;
20
21         ++sequence; // Move to the next file name
22
23         name = "tmp";
24
25         // Put in the squence digit
26         name += static_cast<char>(sequence + '0');
27
28         return(name);
29 }
30
31 int main()
32 {
33         std::string name1 = tmp_name();
34
35         std::cout <<"Name1: " << name1 << '\n';
36         return(0);
37 }
```

(راهنمایی ۳۶۱، جواب ۳۶)

[1] Grandson of Weird Names

برنامه ۶۱: مرور کردن یک واژه نامه به آهستگی[1]

من این برنامه را وقتی که دانشجوی سال سوم Caltech بودم نوشتم (ابتدائا به پاسکال نوشته شده بود). هجی من ضعیف بود برای همین نیاز به چیزی داشتم که با کمک آن بتوانم کلمات را در واژه نامه پیدا کنم.

تصمیم گرفتم برنامه بنویسم که یک واژه نامه را بصورت یک درخت دودویی بخواند و کلمات را در آن بیابد.

درخت های دودویی، ساختمان داده ای کارا هستند، ولی این برنامه زمان بسیار زیادی برای اجرا صرف می کند. چرا؟

```
1 /************************************************
2 * find_word -- find a word in the dictionary. *
3 * *
4 * Usage: *
5 * find_word <word-start> [<word-start>...] *
6 ************************************************/
7 #include <iostream>
8 #include <fstream>
9 #include <iomanip>
10 #include <cctype>
11 #include <cstring>
12 #include <cstdlib>
13
14 /************************************************
15 * tree -- A simple binary tree class *
16 * *
17 * Member functions: *
18 * enter -- Add an entry to the tree *
19 * find -- See if an entry is in the tree. *
20 ************************************************/
21 class tree
22 {
23         private:
24                 // The basic node of a tree
25                 class node {
26                         private:
27                                 // tree to the right
28                                 node *right;
29
30                                 // tree to the left
31                                 node *left;
32                         public:
33                                 // data for this tree
34                                 char *data;
35
36                         public:
37                                 node() :
38                                         right(NULL), left(NULL),
39                                         data(NULL) {}
40                                 // Destructor defaults
41                         private:
42                                 // No copy constructor
43                                 node(const node &);
44
45                                 // No assignment operator
46                                 node & operator = (const node &);
```

[1] Looking Through a Dictionary Slowly

```
47
48                                      // Let tree manipulate our data
49                                      friend class tree;
50
51              };
52
53              // the top of the tree
54              node *root;
55
56              // Enter a new node into a tree or
57              // sub-tree
58              void enter_one(
59                      // Node of sub-tree to look at
60                      node *&node,
61
62                      // Word to add
63                      const char *const data
64              );
65
66              // Find an item in the tree
67              void find_one(
68                      // Prefix to search for
69                      const char start[],
70
71                      // Node to start search
72                      const node *const node,
73
74                      // Keep looking flag
75                      const bool look
76              );
77      public:
78              tree(void) { root = NULL;}
79              // Destructor defaults
80      private:
81              // No copy constructor
82              tree(const tree &);
83
84              // No assignment operator
85              tree & operator = (const tree &);
86
87      public:
88              // Add a new data to our tree
89              void enter(
90                      // Data to add
91                      const char *const data
92              ) {
93                      enter_one(root, data);
94              }
95
96              // Find all words that start
97              // with the given prefix
98              void find(
99                      const char start[] // Starting string
100             )
101             {
102                     find_one(start, root, true);
103             }
104 };
105
106 /************************************************
107 * tree::enter_one -- enter a data into *
108 * the tree *
```

```
109 ************************************************/
110 void tree::enter_one(
111         node *&new_node, // Sub-tree to look at
112         const char *const data // Word to add
113 )
114 {
115         int result; // result of strcmp
116
117         // see if we have reached the end
118         if (new_node == NULL) {
119                 new_node = new node;
120
121                 new_node->left = NULL;
122                 new_node->right = NULL;
123                 new_node->data = strdup(data);
124         }
125
126         result = strcmp(new_node->data, data);
127         if (result == 0) {
128                 return;
129         }
130
131         if (result < 0)
132                 enter_one(new_node->right, data);
133         else
134                 enter_one(new_node->left, data);
135 }
136
137 /************************************************
138 * tree::find_one -- find words that match this *
139 * one in the tree. *
140 ************************************************/
141 void tree::find_one(
142         const char start[], // Start of the work
143         const node *const top,// Top node
144         const bool look // Keep looking
145 )
146 {
147         if (top == NULL)
148                 return; // short tree
149
150         // Result of checking our prefix
151         // against the word
152         int cmp = strncmp(start,
153         top->data, strlen(start));
154
155         if ((cmp < 0) && (look))
156                 find_one(start, top->left, true);
157         else if ((cmp > 0) && (look))
158                 find_one(start, top->right, true);
159
160         if (cmp != 0)
161                 return;
162
163          /*
164         * We found a string that starts this one.
165         * Keep searching and print things.
166         */
167         find_one(start, top->left, false);
168         std::cout << top->data << '\n';
169         find_one(start, top->right, false);
170 }
```

```
171
172 int main(int argc, char *argv[])
173 {
174       // A tree to hold a set of words
175       tree dict_tree;
176
177       // The dictionary to search
178       std::ifstream dict_file("/usr/dict/words");
179
180       if (dict_file.bad()) {
181                 std::cerr <<
182                 "Error: Unable to open "
183                 "dictionary file\n";
184                 exit (8);
185       }
186
187       /*
188       * Read the dictionary and construct the tree
189       */
190       while (1) {
191                 char line[100]; // Line from the file
192
193                 dict_file.getline(line, sizeof(line));
194
195                 if (dict_file.eof())
196                           break;
197
198                 dict_tree.enter(strdup(line));
199       }
200       /*
201       * Search for each word
202       */
203       while (argc > 1) {
204                 std::cout << "------ " << argv[1] << '\n';
205                 dict_tree.find(argv[1]);
206                 ++argv;
207                 --argc;
208       }
209       return (0);
210 }
```

(راهنمایی ۴۲، جواب ۷۴)

برنامه ۶۲: اعمال نیرو[1]

چه چیز ساده تر از انتساب یک مقدار به دو ثابت و نمایش آن است؟ با این در چیزی به این سادگی هم مشکل وجود دارد. چرا یکی از کدهای پستی اشتباه است؟

```
1 /*************************************************
2 * print_zip -- Print out a couple of zip codes.*
3 *************************************************/
4 #include <iostream>
5 #include <iomanip>
6
7 int main()
8 {
9         // Zip code for San Diego
10        const long int san_diego_zip = 92126;
11
12        // Zip code for Boston
13        const long int boston_zip = 02126;
14
15        std::cout << "San Diego " << std::setw(5) <<
16                  std::setfill('0') <<
17                  san_diego_zip << std::endl;
18
19        std::cout << "Boston " << std::setw(5) <<
20                  std::setfill('0') <<
21                  boston_zip << std::endl;
22
23        return (0);
24 }
```

(راهنمایی ۲۰۶، جواب ۱۵)

قانون اللاین برای کامپیوترها
۱- هیچ چیزی در علوم کامپیوتر، مهم تر از چنگ انداختن به چیزهای بدیهی نیست. ۲- هیچ چیز بدیهی در مورد کامپیوترها وجود ندارد.

[1] Zipping Along

فصل ۵: کد C، شکست کد C[1]

علیرغم تلاش طراحان زبان، هنوز به مقدار زیادی کد C وجود دارد. C زبان خودش است و مشکلات خودش را دارد. در این جا تعداد کمی خطای منحصر بفرد و خاص وجود دارد که فقط یک برنامه نویس C می تواند آنها را مرتکب شود.

برنامه ۶۳: اسم – بازی[2]

این برنامه قرار است نام و نام خانوادگی را با هم ترکیب کرده و آنها را چاپ کند.

یک اجرای نمونه می تواند اینگونه باشد:

```
First: John
Last: Smith
Hello: John Smith
Thank you for using Acme Software
```

ولی این برنامه واقعا چکار می کند؟

```
1 /************************************************
2 * Greetings -- Ask the user for his first *
3 * name and his last name. *
4 * Then issue a greeting. *
5 ************************************************/
6 #include <stdio.h>
7 #include <string.h>
8 int main()
9 {
10        char first[100]; /* The first name */
11        char last[100]; /* The last name */
12        char full_name[201];/* The full name */
13
14        /* Get the first name */
15        printf("First: ");
16        fgets(first, sizeof(first), stdin);
17
18        /* Get the last name */
19        printf("Last: ");
20        fgets(last, sizeof(last), stdin);
21
22        /* Make full_name = "<first> <last>" */
23        strcpy(full_name, first);
24        strcat(full_name, " ");
25        strcat(full_name, last);
26
27        /* Greet the user by name */
28        printf("Hello %s\n", full_name);
29        printf("Thank you for "
30                "using Acme Software.\n");
31        return (0);
32 }
```

(راهنمایی ۳۴۰، جواب ۳۳)

[1] C Code, C Code Break
[2] Name Game

برنامه ۶۴: π در چشمان شما[1]

فایل math.h ثابت M_PI را تعریف می کند. وقتی این ثابت را چاپ می کنیم، چه مقداری را دریافت می داریم؟

```
1 /************************************************
2 * PI -- Test program to see verify that *
3 * the value of "pi" in math.h is *
4 * correct. *
5 ************************************************/
6 /* math.h defines M_PI */
7 #include <math.h>
8 #include <stdio.h>
9
10 int main()
11 {
12         printf("pi is %d\n", M_PI);
13         return (0);
14 }
```

(راهنمایی ۱۹۸، جواب ۱۰)

یک نفر در Caltech، برنامه ای نوشته بود تا وقتی شما وارد سیستم می شوید، یک پیغام خوش آمد گویی را نمایش دهد. آن، برنامه باهوشی بود؛ قسمتی از برنامه، account نویسنده برنامه را مورد بررسی قرار می داد که آیا نسخه جدیدتری از برنامه آماده است یا نه. اگر بود، برنامه، خودش را با نسخه جدیدتر جایگزین می ساخت.

روزی نویسنده برنامه، فارغ التحصیل شد و account او پاک شد. برنامه این امر را بعنوان یک خطا در نظر گرفت و یک پیغام را نمایش داد:

?LGNPFB Program fall down and go boom.

[1] π in Your Eye

برنامه ۶۵: جنون آنی[1]

برخی مواقع یک اسم فایل ساختگی برگردانده می شود. برخی اوقات از کار می افتد. چرا؟

```
1 /************************************************
2 * full_test -- Test the full_name function *
3 ************************************************/
4 #define PATH "/usr/tmp"
5
6 /************************************************
7 * full_name -- Given the name of a file, *
8 * return a full path name. *
9 * *
10 * Returns: Absolute path to the file name. *
11 ************************************************/
12 char *full_name(
13         const char name[] /* Base file name */
14 )
15 {
16         /* Full file name */
17         static char file_name[100];
18
19         strcpy(file_name, PATH);
20         strcat(file_name, '/');
21         strcat(file_name, name);
22         return (file_name);
23 }
24
25 int main()
26 {
27         /* Test the full_name funtion */
28         printf("Full name is %s\n",
29         full_name("data"));
30         return (0);
31 }
```

(راهنمایی ۳۲۰، جواب ۴۱)

[1] Temporary Insanity

برنامه ۶۶: ذخیره در هیچ جا[1]

برنامه نویس تصمیم گرفت تا ورودی/خروجی بافر شده را از طریق افزایش اندازه بافر، تسریع بخشد. بطور عادی، این امر باید سرعت را افزایش دهد ولی در این مورد خاص، چیزهای عجیب و غریبی رخ می دهد. چرا؟

```
1 /*************************************************
2 * buffer demo. Show how big buffers can speed *
3 * up I/O. *
4 *************************************************/
5 #include <stdio.h>
6
7 /* Nice big buffer */
8 #define BUF_SIZE (50 * 1024)
9
10 /*************************************************
11 * print_stuff -- Print a bunch of stuff in a *
12 * big buffer. *
13 *************************************************/
14 void print_stuff(void)
15 {
16         // Buffer to hold the data
17         char buffer[BUF_SIZE];
18
19         // Printing counter.
20         int i;
21
22         /* Things go much faster with this */
23         setbuf(stdout, buffer);
24
25         for (i = 0; i < 10; ++i)
26                 printf("Hello world\n");
27         }
28
29
30 int main()
31 {
32         print_stuff();
33         printf("That's all\n");
34         return (0);
35 }
```

(راهنمایی ۷۴، جواب ۸۳)

[1] Buffer to Nowhere

برنامه ۶۷: بیایید "مشکل را پنهان کن" بازی کنیم[1]

برنامه زیر، با یک خطای تقسیم اعشاری بر روی یونیکس مواجه می شود. این امر خیلی سردرگم کننده است چون ما هیچ عملیات اعشاری انجام نمی دهیم.

برای این که مشکل را پیدا کنیم، تعداد کمی دستور printf در برنامه گنجاندیم و فهمیدیم که مشکل جایی قبل از فراخوانی تابع است. این را از آنجا می گوئیم که هیچ وقت پیغام "starting" مشاهده نمی شود.

```
1 /************************************************
2 * Compute a simple average. Because this *
3 * takes a long time (?) we output some *
4 * chatter as we progress through the system. *
5 ************************************************/
6 #include <stdio.h>
7
8 /************************************************
9 * average -- Compute the average given the *
10 * total of the series and the number *
11 * of items in the series. *
12 * *
13 * Returns: *
14 * The average. *
15 ************************************************/
16 int average(
17         const int total,// The total of the series
18         const int count // The number of items
19 )
20 {
21         return (total/count);
22 }
23
24 int main()
25 {
26          int ave; // Average of the number
27
28         printf("Starting....");
29         ave = average(32, 0);
30         printf("..done\n");
31
32         printf("The answer is %d\n", ave);
33         return (0);
34 }
```

(راهنمایی ۱۰۸، جواب ۶۸)

[1] Let's Play "Hide the Problem"

برنامه ۶۸: محاسبه اشتباه[1]

قرار است یک ماشین حساب چهار – کاره بسازیم. کاربر باید یک عملگر و یک عدد را وارد کند و برنامه شروع به کار می کند. مثلا:

Enter operator and value: +10
Total: 10

ولی کارها آنطور که انتظار می رورد، پیش نمی رود.

```
1 /************************************************
2 * calc -- Simple 4 function calculator. *
3 * *
4 * Usage: *
5 * $ calc *
6 * Enter operator and value: + 5 *
7 * *
8 * At the end of each operation the accumulated *
9 * results are printed. *
10 ************************************************/
11 #include <stdio.h>
12 int main() {
13         char oper; /* Operator for our calculator */
14         int result;/* Current result */
15         int value; /* Value for the operation */
16
17         result = 0;
18         while (1)
19         {
20                 char line[100]; // Line from the user
21                 printf("Enter operator and value:");
22
23                 fgets(line, sizeof(line), stdin);
24                 sscanf(line, "%c %d", oper, value);
25
26                 switch (oper) {
27                         case '+':
28                                 result += value; break;
29                         case '-':
30                                 result -= value; break;
31                         case '*':
32                                 result *= value; break;
33                         case '/':
34                                 if (value == 0)
35                                         printf("Divide by 0 error\n");
36                                 else
37                                         result /= value;
38                                 break;
39                         case 'q':
40                                 exit (0);
41                         default:
42                                 printf("Bad operator entered\n"); break;
43                 }
44                 printf("Total: %d\n", result);
45         }
46 }
```

(راهنمایی ۷۳، جواب ۹۵)

[1] Miscalculating

برنامه ۶۹: مشکل جمع[1]

این برنامه طراحی شده است تا سه عدد 1, 2, 3 را با هم جمع کند. ولی وقتی آنرا اجرا می کنیم، نتیجه بصورت زیر است:

Sum is 1343432864

چرا؟

```
1 /*************************************************
2 * sum_test -- Test the sum function *
3 *************************************************/
4 #include <stdio.h>
5
6 /*************************************************
7 * sum -- Sum up three numbers *
8 * *
9 * Returns: The sum of the numbers. *
10 *************************************************/
11 int sum(i1, i2, i3)
12 {
13         int i1; /* The first number */
14         int i2; /* The second number */
15         int i3; /* The third number */
16
17         return (i1 + i2 + i3);
18 }
19
20 int main()
21 {
22         printf("Sum is %d\n", sum(1, 2, 3));
23         return (0);
24 }
```

(راهنمایی ۶۹، جواب ۹۴)

[1] Sum Problem

برنامه ۷۰: دوی ساده[1]

چرا ۵۹۸۶ = ۲ + ۲؟

```
1 /************************************************
2 * two_plus_two -- So what is 2+2 anyway? *
3 ************************************************/
4 #include <stdio.h>
5
6 int main()
7 {
8         /* Result of the addition */
9         int answer = 2 + 2;
10
11        printf("The answer is %d\n");
12        return (0);
13 }
```

(راهنمایی ۱۶۴، جواب ۸۵)

در پایین چکهای بانکی شما، یک سری عدد وجود دارد که نشانگر شماره بانک و شماره حساب شما می باشد. کلاهبرداری، با پنج دلار، یک حساب در نیویورک باز کرد. سپس چکهای خودش را درست کرد. آنها مثل چکهای واقعی بودند بجز اینکه شماره بانک تغییر کرده بود بطوری که نشانگر بانکی در لوس آنجلس بود.

او سپس یک حساب دیگر در نیویورک باز کرد و یک چک ۱۰۰۰۰ دلاری بعنوان موجودی اولیه در آن ریخت. چک درون دستگاه مرتب سازی اتوماتیک رفت و کامپیوتر با مشاهده شماره بانک لوس آنجلس، چک را به آنجا فرستاد. بانک لوس آنجلس متوجه شد که این چک متعلق به آنها نیست لذا آنرا برای دفتر تسویه حساب به نیویورک پس فرستاد. چک دوباره وارد دستگاه مرتب سازی اتوماتیک شد، کامپیوتر شماره بانک لوس آنجلس را دید و آنرا به آنجا فرستاد.

چک اکنون در یک چرخه بی پایان بین لوس آنجلس و نیویورک در گردش بود. در این اثنا، کلاهبردار به بانک رفت و همه پولش را خواست. کارمند به آخرین واریز نگاه کرد و دید که مربوط به دو هفته پیش است و فکر کرد که چک تسویه شده است. چون کلا دو روز طول می کشد تا یک چک نیویورک به بانک مقصد برسد. بنابراین کارمند به کلاهبردار، ۱۰۰۰۰ دلار داد و او ناپدید شد.

چندین هفته بعد، چک آنقدر داغان شده بود که دیگر نمی توانست داخل دستگاه مرتب سازی اتوماتیک قرار بگیرد. بنابراین به طور دستی مرتب شد و به بانک مربوطه فرستاده شد.

[1] Two Simple

برنامه ۷۱: ناهمگام[1]

قرار است یک ماشین حساب چهارکاره بسازیم. کاربر باید یک عملگر و یک عدد وارد نماید و ماشین حساب شروع به کار می کند. مثلا:

Enter operator and value: +10
Total: 10

ولی کارها آنطور که انتظار می رود پیش نمی رود.

```
1 /************************************************
2 * calc -- Simple 4 function calculator. *
3 * *
4 * Usage: *
5 * $ calc *
6 * Enter operator and value: + 5 *
7 * *
8 * At the end of each operation the acculated *
9 * results are printed. *
10 ************************************************/
11 #include <stdio.h>
12 #include <stdlib.h>
13 int main() {
14         char oper; /* Operator for our calculator */
15         int result;/* Current result */
16         int value; /* Value for the operation */
17
18         result = 0;
19         while (1)
20         {
21                 printf("Enter operator and value:");
22                 scanf("%c %d", &oper, &value);
23
24                 switch (oper) {
25                         case '+':
26                                 result += value;
27                                 break;
28                         case '-':
29                                 result -= value;
30                                 break;
31                         case '*':
32                                 result *= value;
33                                 break;
34                         case '/':
35                                 if (value == 0)
36                                         printf("Divide by 0 error\n");
37                                 else
38                                         result /= value;
39                                  break;
40                         case 'q':
41                                 exit (0);
42                         default:
43                                 printf("Bad operator entered\n"); break;
44                 }
45                 printf("Total: %d\n", result);
46         }
47 }
```

(راهنمایی ۲۲۴، جواب ۲۸)

[1] Unsynchronized

برنامه ۷۲: بینش بی انتها[1]

این برنامه ساده برای کپی کردن ورودی استاندارد به خروجی استاندارد است. این یکی از اولین برنامه های مرتبط با I/O است که یک دانشجو می نویسد.

```
1 /*************************************************
2 * copy -- Copy stdin to stdout. *
3 *************************************************/
4 #include <stdio.h>
5
6 int main()
7 {
8         // Character to copy
9         char ch;
10
11        while ((ch = getchar()) != EOF)
12        {
13                putchar(ch);
14        }
15        return (0);
16 }
```

(راهنمایی ۱۵، جواب ۶۳)

دو راه برای طراحی نرم افزار وجود دارد. یک راه این است که آنرا آنقدر ساده سازیم که آشکارا هیچ ایرادی نداشته باشد و راه دیگر این است که آنرا آنقدر پیچیده سازیم که هیچ ایراد آشکاری نداشته باشد.
س. ا. ر. هوار

[1] No End in Sight

فصل ۶: شکست زودرس[1]

پیش پردازنده C++ انعطاف پذیری بیشتری به زبان می دهد. همچنین راه های جدیدی برای ارتکاب اشتباه پیش روی شما قرار می دهد.

برنامه ۷۳: بیهوده[2]

انواع داده sam و joe چیستند؟

```
1 /***********************************************
2 * Toy program that declares two variables *
3 ***********************************************
4 #define CHAR_PTR char *
5
6 int main()
7 {
8         CHAR_PTR sam, joe;
9
10        return (0);
11 }
```

(راهنمایی ۲۹۸، جواب ۷۸)

من روی اولین چاقوی فشار آب کار می کردم. آن دستگاهی بود که کف کفش های تنیس را با فشار بالایی از آب می برید. از آنجا که آن، اولین نمونه از نوع خود بود، زمان بسیاری را برای میزان سازی آن صرف کردیم، حدود یک سال. ما با شرکت سازنده کفش تنیس که قرار بود آنرا بخرد، قراردادی داشتیم. آنها به ما مواد خام رایگان برای آزمایش می دادند اگر که تکه های بریده شده را به آنها برمی گرداندیم.

ما حدود یکسال آزمایش کردیم. چون می خواستیم همیشه نتایج همسان داشته باشیم، تقریبا همیشه از یک آزمون استفاده می کردیم: ۹ راست. ما با وظیفه شناسی، قطعات بریده شده را بسته بندی می کردیم و به سازنده کفش های تنیس باز می گرداندیم که آنها بتوانند از آن تکه ها کفش بسازند یا حداقل ما اینطور فکر می کردیم.

حدود یک هفته قبل از اینکه دستگاه را تحویل دهیم، کسی از کارخانه کفش تنیس با ما تماس گرفت.

کارخانه کفش تنیس: "آیا شما همان افرادی هستید که مرتب به ما ۹ راست را می فرستید؟"
ما: "بله"

- بالاخره شما را پیدا کردم. برای یکسال دنبالتان بودم. هیچ اطلاعاتی از خرید قطعات بریده شده وجود نداشت و یافتن شما خیلی سخت بود.
- مشکلی پیش آمده؟
- بله. آیا متوجه نشده اید که شما ۱۰۰۰۰ راست نه به ما تحویل داده اید و هیچ لنگه چپی نفرستاده اید؟

[1] Premature Breakage
[2] Pointless

برنامه ۷۴: خطای فاحش[1]

چرا برنامه زیر در خط ۱۶ یک خطای نحوی را گزارش می دهد. خط ۱۶ چه مشکلی دارد؟

```
1 /************************************************
2 * gross -- Print out a table of 1 to 10 gross. *
3 ************************************************/
4 // A Gross is a dozen - dozen
5 #define GROSS (12 ** 2)
6
7 #include <iostream>
8
9 int main()
10 {
11          int i; // Index into the table
12
13          for (i = 1; i <= 10; ++i)
14          {
15                    std::cout << i << " gross is " <<
16                    (GROSS * i) << '\n';
17          }
18
19          return (0);
20 }
```

(راهنمایی ۲۷۵، جواب ۷۹)

دو راه برای نوشتن برنامه های بدون اشکال وجود دارد ولی فقط سومی درست کار می کند.

[1] Gross Error

برنامه ۷۵: خروج اضطراری[1]

ماکروی ABORT طراحی شده است تا یک پیغام خطا صادر کرده و خارج شود. برنامه باید وقتی اشتباهی پیش می آید، قطع شود.

وقتی خطایی پیش می آید برنامه خارج می شود. وقتی خطایی نداریم هم برنامه خارج می شود. درحقیقت، در هر حال برنامه خارج می شود.

چرا؟

```
1 /************************************************
2 * Test the square_root function. *
3 ************************************************/
4 #include <iostream>
5 #include <math.h>
6
7 /************************************************
8 * ABORT -- print an error message and abort. *
9 ************************************************/
10 #define ABORT(msg) \
11         std::cerr << msg << std::endl;exit(8);
12 /************************************************
13 * square_root -- Find the square root of the *
14 * value. *
15 * *
16 * Returns: *
17 * The square root. *
18 ************************************************/
19 static int square_root(
20         const int value
22 ) {
23         if (value < 0)
24         ABORT("Illegal square root");
25
26         return (int(sqrt(float(value))));
27 }
28
29 int main() {
30         int square; // A number that's square
31         int root; // The square root of the number
32
33         square = 5 * 5;
34         root = square_root(square);
35
36         std::cout << "Answer is: " << root << '\n';
37         return (0);
38 }
```

(راهنمایی ۳۳، جواب ۱۰۵)

[1] Quick Exit

برنامه ۷۶: مشکل دو چندان[1]

ماکروی DOUBLE طراحی شده است تا مقدار آرگومان خود را دوبرابر کند. برنامه آزمون، مقادیر DOUBLE اعداد ۱ تا ۵ را می نویسد. ولی یک جای کار ایراد دارد. مشکل چیست؟

```
1 /*************************************************
2 * Double -- Print double table. *
3 * *
4 * Print the numbers 1 through 5 and their *
5 * doubles. *
6 *************************************************/
7 #include <iostream>
8
9 /*************************************************
10 * DOUBLE -- Given a number return its double. *
11 *************************************************/
12 #define DOUBLE(x) (x * 2)
13
14 int main()
15 {
16         int i; // Number to print and to double
17
18         for (i = 0; i < 5; ++i) {
19                 std::cout << "The double of " << i+1 <<
20                         " is " << DOUBLE(i+1) << std::endl;
21         }
22
23         return (0);
24 }
```

(راهنمایی ۱۳۳، جواب ۴۶)

زبان برنامه نویسی C – زبانی که انعطاف پذیری زبان اسمبلی و قدرت زبان اسمبلی را درهم می آمیزد.

[1] Double Trouble

برنامه ۷۷: بی مقدار[1]

برنامه زیر کامپایل نمی شود چون value تعریف نشده است. ما هیچ وقت از متغیر value استفاده نمی کنیم، پس مشکل چیست؟

```
1 /*************************************************
2 * double -- Print a double table for the *
3 * numbers 1 through 10. *
4 *************************************************/
5 #include <iostream>
6
7 /*************************************************
8 * DOUBLE -- Macro to double the value of a *
9 * number. *
10 *************************************************/
11 #define DOUBLE (value) ((value) + (value))
12
13 int main()
14 {
15         // Counter for the double list
16         int counter;
17
18         for (counter = 1; counter <= 10; ++counter)
19         {
20                 std::cout << "Twice " << counter << " is " <<
21                 DOUBLE(counter) << '\n';
22         }
23
24         return (0);
25 }
```

(راهنمایی ۱۱۸، جواب ۱۱۳)

[1] No Value

برنامه ۷۸: محدوده خطا[1]

اگر یک کاغذ با پهنای ۸,۵ اینچ داشته باشیم و از ۱ اینچ برای محدوده ها (۰,۵ اینچ برای هر طرف) استفاده کنیم، چقدر فضای قابل استفاده باقی گذاشته ایم؟ همه می دانند که جواب ۷,۵ اینچ است. ولی این برنامه از زاویه دیگری به قضایا می نگرد. چه شده است؟

```
1 /************************************************
2 * paper_size -- Find the usable width on *
3 * a page. *
4 ************************************************/
5 #define PAPER_WIDTH 8.5; // Width of the page
6 #define MARGIN 1.0; // Total margins
7 // Usable space on the page
8 #define USABLE PAPER_WIDTH -MARGIN;
9
10 #include <iostream>
11
12 int main()
13 {
14         // The usable width
15         double text_width = USABLE;
16
17         std::cout << "Text width is " <<
18                 text_width << '\n';
19         return (0);
20 }
```

(راهنمایی ۴۵، جواب ۸۲)

در اوقات فراغتم، من بازی Adventure را روی کامپیوتر شرکت نصب کرده بودم و ساعاتی را به بازی کردن می گذراندم. روزی مدیرم مرا به دفتر خود فراخواند.

او پرسید: "آیا شما روی سیستم، Adventure نصب کرده اید؟"

جواب دادم: "من آن را در اوقات فراغتم بازی می کردم."

او مرا خاطرجمع کرد که: "آه، من از شما انتقاد نمی کنم. حقیقت امر این است که می خواهم از شما قدردانی کنم. از موقعی که این پروژه شروع شده است، بیل (مسئول بازاریابی) هرروز این جا می آمد. هر روز او وارد می شود، با نرم افزار کار می کند و اصرار به برخی تغییرات در برنامه دارد. ولی در هفته گذشته، تمام وقت خود را به بازی با Adventure گذراند و وقتی برایش نماند که تقاضای تغییرات بکند. من می خواستم از شما به خاطر اینکه شر او را از سر من کم کرده اید، تشکر کنم.

[1] Margin of Error

برنامه ۷۹: تقلب نکردن[1]

C++ عملگر توان ندارد، لذا ما ماکرویی برای محاسبه X^2 تعریف می کنیم. تصمیم گرفتیم که این ماکرو را با مجذور اعداد ۱ تا ۱۰ آزمایش کنیم. ولی واقعا داریم چه کار می کنیم؟

```
1 /************************************************
2 * Print out the square of the numbers *
3 * from 1 to 10. *
4 ************************************************/
5 #include <iostream>
6
7 /********************************** ************
8 * macro to square a number. *
9 ************************************************/
10 #define SQR(x) ((x) * (x))
11
12 int main()
13 {
14         int number; // The number we are squaring
15
16         number = 1;
17
18         while (number <= 10) {
19                 std::cout << number << " squared is " <<
20                 SQR(++number) << std::endl;
21         }
22
23         return (0);
24 }
```

(راهنمایی ۲۰۰، جواب ۸۸)

[1] Square Deal

برنامه ۸۰: بمباران منطقه ای[1]

می خواهیم که مساحت یک مستطیل را حساب کنیم. ما طول را در دو قسمت و عرض را داریم. ولی چرا ماکروی زیر، مساحت غلط را برمی گرداند؟

```
1 /*************************************************
2 * Find the area of a rectangle. The top of *
3 * the rectangle consists of two parts, *
4 * cleverly called PART1 and PART2. *
5 * The side is called SIDE. *
6 * *
7 * So our rectangle looks like: *
8 * <- TOP_PART1 ->|<-- TOP_PART2 -> | *
9 * +------------------------------+ ^ *
10 * | | | *
11 * | | | *
12 * | | | SIDE *
13 * | | | *
14 * | | | *
15 * +------------------------------+ V *
16 *************************************************/
17
18 // First leg of top is 37 feet
19 #define TOP_PART1 37
20
21 // Second part of the top is 33 feet
22 #define TOP_PART2 33
23
24 // Total top size
25 #define TOP_TOTAL TOP_PART1 + TOP_PART2
26
27 #define SIDE 10 // 10 Feet on a side
28
29 // Area of the rectangle
30 #define AREA TOP_TOTAL * SIDE
31
32 #include <iostream>
33
34 int main() {
35         std::cout << "The area is " <<
36         AREA << std::endl;
37         return (0);
38 }
```

(راهنمایی ۲۸، جواب ۲۹)

[1] Area Bombing

فصل ۷: کلاس هایی بدون کلاس[1]

وقتی بیارنه استروستراپ، C++ را ابداع کرد، نه تنها یک زبان برنامه نویسی بزرگ ایجاد کرد بلکه زبان بزرگی ایجاد کرد که به برنامه نویس، قدرت فوق العاده می داد. او همچنین یک مجموعه جدید از روشهای ارتکاب اشتباه را به برنامه نویس هدیه کرد. در سایه تلاش های او، تمام برنامه های این فصل امکانپذیر شدند.

برنامه ۸۱: سپاس برای حافظه[2]

چرا این برنامه، حافظه کم می آورد؟

```
1 /**************************************************
2 * play with a variable size stack class. *
3 **************************************************/
4
5 /**************************************************
6 * stack -- Simple stack class *
7 * *
8 * Member functions: *
9 * push -- Push data on to the stack *
10 * pop -- remove an item from the stack. *
11 **************************************************/
12 class stack
13 {
14        private:
15                int *data; // The data
16                const int size; // The size of the data
17
18                // Number of items in the data
19                int count;
20        public:
21                // Create the stack
22                stack(
23                        // Max size of the stack
24                        const int _size
25                ):size(_size), count(0)
26                {
27                        data = new int[size];
28                }
29                ~stack(void) {}
30        private:
31                // No copy constructor
32                stack(const stack &);
33
34                // No assignment operator
35                stack & operator = (const stack &);
36        public:
37                // Push something on the stack
38                void push(
39                        // Value to put on stack
40                        const int value
41                )
42                {
43                        data[count] = value;
44                        ++count;
45                }
```

[1] Classes with No Class
[2] Thanks for the Memory

```
46                  // Remove an item from the stack
47                  int pop(void)
48                  {
49                          --count;
50                          return (data[count]);
51                  }
52 };
53
54 int main()
55 {
56          stack a_stack(30);
57
58          a_stack.push(1);
59          a_stack.push(3);
60          a_stack.push(5);
61          a_stack.push(7);
62          return (0);
63 }
```

(راهنمایی ۵۶، جواب ۳۲)

برنامه ۸۲: مورد آرایه ناپدید شونده[1]

ما یک کلاس ساده آرایه و یک روتین ساده تر برای آزمون داریم. ولی بطریقی حافظه خراب می شود.

```
1 /*************************************************
2 * var_array -- Test variable length array *
3 * class. *
4 *************************************************/
5 #include <memory.h>
6
7 /*************************************************
8 * var_array -- Variable length array *
9 * *
10 * Member functions: *
11 * operator [] -- Return a reference to *
12 * the item in the array. *
13 *************************************************/
14
15 class var_array
16 {
17         private:
18                 int *data; // The data
19                 const int size; // The size of the data
20         public:
21                 // Create the var_array
22                 var_array(const int _size):
23                 size(_size)
24                 {
25                         data = new int[size];
26                         memset(data, '\0',
27                         size * sizeof(int));
28                 }
29                 // Destroy the var_array
30                 ~var_array(void) {
31                         delete []data;
32                 }
33         public:
34                 // Get an item in the array
35                 int &operator [] (
36                         // Index into the array
37                         const unsigned index
38                 )
39                 {
40                         return (data[index]);
41                 }
42 };
43
44 /*************************************************
45 * store_it -- Store data in the var_array *
46 *************************************************/
47 static void store_it(
48         // Array to use for storage
49         var_array test_array
50 )
51 {
52         test_array[1] = 1;
53         test_array[3] = 3;
54         test_array[5] = 5;
```

[1] The Case of the Disappearing Array

```
55          test_array[7] = 7;
56 }
57 int main()
58 {
59          var_array test_array(30);
60
61          store_it(test_array);
62          return (0);
63 }
```

(راهنمایی ۱۸۹، جواب ۵۹)

قانون مستندسازی اللاین

۹۰ درصد اوقات، مستندسازی وجود ندارد. در ۱۰ درصد باقیمانده، ۹ درصد اوقات، مستندسازی برای یک نسخه اولیه نرم افزار است و لذا بدون استفاده می باشد. در ۱ درصد مواقع، شما مستندسازی و نسخه درستی از آن را دارید ولی به ژاپنی نوشته شده است.

من این طنز را به رفیقی در موتورولا گفتم و او برای چند دقیقه خندید. سپس یک راهنمای فرترن هیتاچی را بیرون آورد که به ژاپنی نوشته شده بود.

برنامه ۸۳: خروجی وحشی[1]

یک دانشجوی C++ می خواست ببیند که سازنده[2] ها و مخرب[3] ها چگونه فراخوانی می شوند، لذا برنامه زیر را نوشت. او چیزی بیش از آنچه که توقع داشت فراگرفت. مشکل چیست؟

```
1 /************************************************
2 * Class tester. Test constructor / destructor*
3 * calling. *
4 ************************************************/
5 #include <iostream>
6
7 /************************************************
8 * tester -- Class that tells the world when *
9 * it's created and destroyed. *
10 ************************************************/
11 class tester {
12         public:
13                 tester(void) {
14                         std::cout <<
15                         "tester::tester() called\n";
16                 }
17                 ~tester(void) {
18                         std::cout <<
19                         "tester::~tester() called\n";
20                 }
21 };
22
23 static tester a_var; // Variable to test with
24
25 int main()
26 {
27         std::cout << "In main\n";
28         return (0);
29 }
```

(راهنمایی ۱۵۷، جواب ۱۱۱)

[1] Wild Output
[2] Constructor
[3] Destructor

برنامه ۸۴: پروژه سازندگی[1]

دانش آموز می خواست ببیند که چه وقتی سازنده کپی[2] و عملگر = فراخوانده می شوند، لذا برنامه زیر را نوشت. ولی از دیدن نتایج شگفت زده شد. چه شده است؟

```
1 #include <iostream>
2 /************************************************
3 * trouble -- A class designed to store a *
4 * single data item. *
5 * *
6 * Member function: *
7 * put -- put something in the class *
8 * get -- get an item from the class *
9 ************************************************/
10 class trouble {
11         private:
12                 int data; // An item to be stored
13         public:
14                 trouble(void) { data = 0; }
15
16                 trouble(const trouble &i_trouble) {
17                         std::cout << "Copy Constructor called\n";
18                         *this = i_trouble;
19                 }
20                 trouble operator = (const trouble &i_trouble) {
21                         std::cout << "= operator called\n";
22                         data = i_trouble.data;
23                         return (*this);
24                 }
25         public:
26                 // Put an item in the class
27                 void put(const int value) {
28                         data = value;
29                 }
30                 // Get an item from the class
31                 int get(void) {
32                         return (data);
33                 }
34 };
35
36 int main() {
37         trouble first; // A place to put an item
38         first.put(99);
39
40         trouble second(first); // A copy of this space
41
42         std::cout << "Second.get " << second.get() << '\n';
43
44         return (0);
45 }
```

(راهنمایی ۲۹۱، جواب ۱۰۹)

[1] Construction Project
[2] Copy Constructor

برنامه ۸۵: صف بندی طولانی[1]

این برنامه یک کلاس ساده و خوش ترکیب queue درست می کند. ولی وقتی از آن استفاده می کنیم، حافظه خراب می شود. چرا؟

```
1 /*************************************************
2 * test the variable length queue class. *
3 *************************************************/
4 #include <iostream>
5
6 /*************************************************
7 * queue -- Variable length queue class. *
8 * *
9 * Member functions: *
10 * queue(size) -- Create a queue that can *
11 * hold up to size elements. *
12 * *
13 * get -- Return an item from the queue. *
14 * (Elements are gotten in First *
15 * In First Out (FIFO) order.) *
16 * put -- Add an item to the queue. *
17 * *
18 * WARNING: No safety check is made to make *
19 * sure something is in the queue before *
20 * it is removed. *
21 *************************************************/
22 class queue
23 {
24         private:
25                 t *data; // The data
26                 t in_index; // Input index
27                 t out_index; // Output index
28                 t size; // # items in the queue
29
30                 Copy data from another queue to me.
31                 void copy_me(
32                         // Stack to copy from
33                         const queue &other
34                 )
35                 {
36                         int i; // Current element
37
38                         for (i = 0; i < size; ++i) {
39                                 data[i] = other.data[i];
40                         }
41                 }
42
43                 // Inc_index -- Increment an
44                 // index with wrapping
45                 void inc_index(int &index)
46                 {
47                         ++index;
48                         if (index == size)
49                         {
50                                 // Wrap
51                                 index = 0;
52                         }
53                 }
```

[1] Queueing Up Too Long

```
54
55      public:
56              // Create a queue of the given size
57              queue(const int _size):
58                      in_index(o), out_index(o), size(_size)
59              {
60                      data = new int[size];
61              }
62
63              // Destructor
64              ~queue(void) {}
65
66              // Copy constructor
67              queue(const queue &other):
68              in_index(other.in_index),
69              out_index(other.out_index),
70              size(other.size)
71              {
72                      data = new int[size];
73                      copy_me(other);
74              }
75              // Assignment operator
76              queue & operator = (const queue &other)
77              {
78                      copy_me(other);
79                      return (*this);
80              };
81      public:
82              // Put an item on the queue
83              void put(
84                      // Value to Put on the queue
85                      const int value
86              )
87              {
88                      data[in_index] = value;
89                      inc_index(in_index);
90              }
91              // Return first element from the queue
92              int get(void)
93              {
94                      // Value to return
95                      int value = data[out_index];
96
97                      inc_index(out_index);
98                      return (value);
99              }
100 };
101
102 int main()
103 {
104     // Queue to play around with
105     queue a_queue(30);
106
107     // Loop counter for playing with the queue
108     int i;
109
110     for (i = 0; i < 30; ++i)
111     a_queue.put(i);
112
113     // Create a new queue, same as the current one
114     queue save_queue(20);
115     save_queue = a_queue;
```

```
116
117     std::cout << "Value is " <<
118             a_queue.get() << std::endl;
119
120     std::cout << "Value is " <<
121             a_queue.get() << std::endl;
122
123     std::cout << "Value is " <<
124             a_queue.get() << std::endl;
125
126     std::cout << "Value is " <<
127             a_queue.get() << std::endl;
128
129     return (0);
130 }
```

(راهنمایی ۳۳۴، جواب ۱۴)

برنامه ۸۶: عدم خود – آگاهی[1]

برنامه زیر طراحی شده است تا آرایه ساده ما را تست کند. ولی مشکلی وجود دارد که باعث می شود برنامه بطور غیرمنتظره ای از کار بیفتد.

```
1 /*************************************************
2 * array_test -- Test the use of the array class*
3 *************************************************/
4 #include <iostream>
5
6 /*************************************************
7 * array -- Classic variable length array class.*
8 * *
9 * Member functions: *
10 * operator [] -- Return an item *
11 * in the array. *
12 *************************************************/
13 class array {
14        protected:
15                // Size of the array
16                int size;
17
18                // The array data itself
19                int *data;
20        public:
21                // Constructor.
22                // Set the size of the array
23                // and create data
24                array(const int i_size):
25                        size(i_size),
26                        data(new int[size])
27                {
28                        // Clear the data
29                        memset(data, '\0',
30                                size * sizeof(data[0]));
31                }
32                // Destructor -- Return data to the heap
33                virtual ~array(void)
34                {
35                        delete []data;
36                        data = NULL;
37                }
38                // Copy constructor.
39                // Delete the old data and copy
40                array(const array &old_array)
41                {
42                        delete []data;
43                        data = new int[old_array.size];
44
45                        memcpy(data, old_array.data,
46                                size * sizeof(data[o]));
47                }
48                // operator =.
49                // Delete the old data and copy
50                array & operator = (
51                        const array &old_array)
52                {
53                        delete []data;
```

[1] Lack of Self – Awareness

```
54                              data = new int[old_array.size];
55
56                              memcpy(data, old_array.data,
57                                      size * sizeof(data[0]));
58                              return (*this);
59                      }
60              public:
61                      // Get a reference to an item in the array
62                      int &operator [](const unsigned int item)
63                      {
64                              return data[item];
65                      }
66 };
67
68 /*********************************************
69 * three_more_elements -- *
70 * Copy from_array to to_array and *
71 * put on three more elements. *
72 *********************************************/
73 void three_more_elements(
74         // Original array
75         array to_array,
76
77         // New array with modifications
78         const array &from_array
79 )
80 {
81         to_array = from_array;
82         to_array[10] = 1;
83         to_array[11] = 3;
84         to_array[11] = 5;
85 }
86 int main()
87 {
88         array an_array(30); // Simple test array
89
90         an_array[2] = 2; // Put in an element
91         // Put on a few more
92         three_more_elements(an_array, an_array);
93         return(0);
94 }
```

(راهنمایی ۸، جواب ۷۵)

استثنای استثنایی[1]

این کلاس پشته طراحی شده است تا مستحکم[2] تر باشد و اگر چیز اشتباهی در پشته پیش آمد، استثنایی را ایجاد کند[3]. ولی باز هم این برنامه قطع می شود و درست کار نمی کند. چرا؟

```
1 /*************************************************
2 * stack_test -- Yet another testing of a *
3 * stack class. *
4 *************************************************/
5 #include <iostream>
6
7 /*************************************************
8 * problem -- Class to hold a "problem". Used *
9 * for exception throwing and catching. *
10 * *
11 * Holds a single string which describes the *
12 * error. *
13 *************************************************/
14 class problem
15 {
16         public:
17                 // The reason for the exception
18                 char *what;
19
20                 // Constructor.
21                 // Create stack with messages.
22                 problem(char *_what):what(_what){}
23 };
24
25 // Max data we put in a stack
26 // (private to the stack class)
27 const int MAX_DATA = 100;
28 /*************************************************
29 * stack -- Classic stack. *
30 * *
31 * Member functions: *
32 * push -- Push an item on the stack. *
33 * pop -- Remove an item from the stack. *
34 * *
35 * Exceptions: *
36 * Pushing too much data on a stack or *
37 * removing data from an empty stack *
38 * causes an exception of the "problem" *
39 * class to be thrown. *
40 * *
41 * Also if you don't empty a stack *
42 * before you're finished, an exception *
43 * is thrown. *
44 *************************************************/
45 class stack {
46         private:
47                 // The stack's data
48                 int data[MAX_DATA];
49
50                 // Number of elements
51                 // currently in the stack
```

[1] Exceptional Exception
[2] Robust
[3] Throw an Exception

```
52                      int count;
53
54          public:
55                      // Constructor
56                      stack(void) : count(0) {};
57
58                      // Destructor -- Check for non
59                      ~stack(void)
60                      {
61                              if (count != 0)
62                              {
63                                      throw(
64                                              problem("Stack not empty"));
65                              }
66                      }
67
68                      // Push an item on the stack
69                      void push(
70                              const int what // Item to store
71                      )
72                      {
73                              data[count] = what;
74                              ++count;
75                      }
76                      // Remove an item from the stack
77                      int pop(void)
78                      {
79                              if (count == 0)
80                                      throw(
81                                              problem("Stack underflow"));
82                              --count;
83                              return (data[count]);
84                      }
85 };
86
87 /************************************************
88 * push_three -- Push three items onto a stack *
89 * *
90 * Exceptions: *
91 * If i3 is less than zero, a "problem" *
92 * class exception is thrown. *
93 ************************************************/
94 static void push_three(
95          const int i1, // First value to push
96          const int i2, // Second value to push
97          const int i3 // Third value to push
98 )
99 {
100         // Stack on which to push things
101         stack a_stack;
102
103         a_stack.push(i1);
104         a_stack.push(i2);
105         a_stack.push(i3);
106         If (i3 < 0)
107         throw (problem("Bad data"));
108 }
109
110 int main(void)
111 {
112         try {
113                      push_three(1, 3, -5);
```

```
114     }
115     catch (problem &info) {
116
117             std::cout << "Exception caught: " <<
118                     info.what << std::endl;
119
120             exit (8);
121     }
122     catch (...) {
123             std::cout <<
124                     "Caught strange exception " <<
125                     std::endl;
126
127             exit (9);
128     }
129     std::cout << "Normal exit" << std::endl;
130     return (0);
131 }
```

(راهنمایی ۱۱۰، جواب ۵۵)

برنامه ۸۸: این را بایگانی کن![1]

بخاطر برخی خواسته های غیرمنطقی از برنامه، تابع زیر باید یک FILE را درون یک جریان[2]، کپی کند. چرا نمی تواند کارش را درست انجام دهد؟

```
1 /************************************************
2 * copy -- Copy the input file to the output *
3 * file. *
4 ************************************************/
5 #include <cstdio>
6 #include <iostream>
7 #include <fstream>
8
9 /************************************************
10 * copy_it -- Copy the data *
12 void copy_it(
13         FILE *in_file, // Input file
14         std::ostream &out_file // Output file
15 )
16 {
17         int ch; // Current char
18
19         while (1) {
20                 ch = std::fgetc(in_file);
21                 if (ch == EOF)
22                         break;
23                 out_file << ch;
24         }
25 }
26
27 int main()
28 {
29         // The input file
30         FILE *in_file = std::fopen("in.txt", "r");
31         // The output file
32         std::ofstream out_file("out.txt");
33
34         // Check for errors
35         if (in_file == NULL) {
36                 std::cerr <<
37                         "Error: Could not open input\n";
38                 exit (8);
39         }
40         if (out_file.bad()) {
41         std::cerr <<
42         "Error: Could not open output\n";
43         exit (8);
44         }
45         // Copy data
46         copy_it(in_file, out_file);
47
48         // Finish output file
49         std::fclose(in_file);
50         return (0);
51 }
```

(راهنمایی ۱۰، جواب ۹۹)

[1] File This!
[2] Stream

برنامه ۸۹: اینکه من دارای سوظن شدید هستم لزوما به این معنی نیست که برنامه در تعقیب من نمی باشد[1]

برای تشریح مشکل تابع کتابخانه ای setjmp، من یک کلاس v_string ساختم. کد آزمون این تابع در زیر آمده است. من همیشه دقت می کنم که از خطاهای کمبود حافظه جلوگیری کنم. ولی این برنامه درست کار نمی کند چون این دفعه *خیلی* دقت کرده بودم. چه شده است؟

```
1 /************************************************
2 * Combine strings with a variable length *
3 * string class. *
4 ************************************************/
5 #include <iostream>
6 #include <cstring>
7
8 /************************************************
9 * v_string -- variable length C style string *
10 * *
11 * Member functions: *
12 * set -- set the value of the string. *
13 * get -- get the data from the string. *
14 ************************************************/
15 class v_string
16 {
17         public:
18                 const char *data; // The data
19                 // Default constructor
20                 v_string(): data(NULL)
21                 {}
22                 v_string(const char *const i_data):
23                         data(strdup(i_data))
24                 {}
25                 // Destructor
26                 ~v_string(void)
27                 {
28                         // Note: delete works
29                         // even if data is NULL
30                         delete [] data;
31                         data = NULL;
32                 }
33                 // Copy constructor
34                 v_string(const v_string &old)
35                 {
36                         if (data != NULL)
37                         {
38                                 delete[] data;
39                                 data = NULL;
40                         }
41                         data = strdup(old.data);
42                 }
43                 // operator =
44                 v_string & operator = (
45                         const v_string &old)
46                 {
47                         if (this == &old)
48                         return (*this);
49
50                         if (data != NULL)
51                         {
52                                 delete[] data;
```

[1] Just Because I'm Paranoid Doesn't Mean the Program Isn't Out to Get Me

```
53                                      data = NULL;
54                              }
55                              if (old.data == NULL)
56                              {
57                                      data = NULL;
58                                      return (*this);
59                              }
60
61                              data = strdup(old.data);
62                              return (*this);
63              }
64      public:
65              // Set a value
66              void set(
67                      // New string value
68                      const char *const new_data
69              )
70              {
71                      if (data != NULL)
72                      {
73                              delete [] data;
74                              data = NULL;
75                      }
76                      data = strdup(new_data);
77
78              }
79              // Returns the value of the string
80              const char * const get(void) const
81              {
82                      return (data);
83              }
84 };
85 /************************************************
86 * operator + -- Combine two v_strings *
87 ************************************************/
88 v_string operator + (
89      const v_string &first, // First string
90      const v_string &second // Second string
91 )
92 {
93      char tmp[100]; // Combined string
94
95      strcpy(tmp, first.get());
96      strcat(tmp, second.get());
97
98      // Strings put together
99      v_string together(tmp);
100     return (together);
101 }
102
103 /************************************************
104 * combine -- Combine two strings and *
105 * print the result. *
106 ************************************************/
107 static void combine(
108     const v_string &first, // First string
109     const v_string &second // Second string
110 )
111 {
112     v_string together; // Strings put together
113     together = first + second;
114
```

```
115       std::cout << "Combination " <<
116                 together.get() << '\n';
117 }
118
119 int main()
120 {
121       // Strings to combine
122       v_string first("First:");
123       v_string second("Second");
124       combine(first, second);
125       return (0);
126 }
```

(برنامه ۶۵، جواب ۱۱۵)

برنامه ۹۰: به آسانی ثبت وقایع[1]

برای پیدا کردن یک کمبود حافظه، برنامه نویس باهوش ما، تصمیم گرفت تا اطلاعات واقعه نگاری[2] را با تعریف دوباره توابع سرتاسری درون new و delete قرار دهد. با اینکه C++ اجازه این کار را می دهد، ولی برنامه او درست کار نمی کند. چرا؟

```
1 /************************************************
2 * simple debugging library that overrides the *
3 * standard new and delete operators so that we *
4 * log all results. *
5 ************************************************/
6 #include <iostream>
7 #include <fstream>
8 #include <cstdlib>
9
10 // Define the file to write the log data to
11 std::ofstream log_file("mem.log");
12
13 /************************************************
14 * operator new -- Override the system new so *
15 * that it logs the operation. This is *
16 * useful for debugging. *
17 * *
18 * Note: We have verified that the real new *
19 * calls malloc on this system. *
20 * *
21 * Returns a pointer to the newly created area. *
22 ************************************************/
23 void *operator new(
24         // Size of the memory to allocate
25         const size_t size
26 )
27 {
28         // Result of the malloc
29         void *result = (void *)malloc(size);
30
31         log_file <<
32                 result << " =new(" <<
33                         size << ")" << std::endl;
34
35         return (result);
36 }
37
38 /************************************************
39 * operator delete -- Override the system *
40 * delete to log the operation. This is *
41 * useful for debugging. *
42 * *
43 * Note: We have verified that the real delete *
44 * calls free on this system. *
45 ************************************************/
46 void operator delete(
47         void *data // Data to delete
48 )
49 {
50         log_file << data << " Delete" << std::endl;
51         free (data);
```

[1] It's As Easy As Rolling off a Log
[2] Logging

```
52 }
53
54 // Dummy main
55 int main()
56 {
57         return (0);
58 }
```

(راهنمایی ۲۱۲، جواب ۱۱۰)

قانون زبان های برنامه نویسی پیشرفته: به برنامه نویسان این امکان را بدهید که انگلیسی بنویسند، و در خواهید یافت که برنامه نویسان نمی توانند به انگلیسی بنویسند.

برنامه ۹۱: غلط انباشته شده[1]

در برنامه زیر، یک کلاس خطرناک stack و یک کلاس ایمن تر safe_stack تعریف می کنیم. برنامه آزمون ما یک آرایه از پنج پشته درسته می کند و تعدادی داده آزمون در آن قرار می دهد. برنامه، اندازه پشته را می نویسد. ولی نتایج آنچه که انتظار می رفت نیستند.

```
1 /************************************************
2 * stack_test -- Test the use of the classes *
3 * stack and safe_stack. *
4 ************************************************/
5 #include <iostream>
6
7 // The largest stack we can use
8 // (private to class stack and safe_stack)
9 const int STACK_MAX = 100;
10 /************************************************
11 * stack -- Class to provide a classic stack. *
12 * *
13 * Member functions: *
14 * push -- Push data on to the stack. *
15 * pop -- Return the top item from the *
16 * stack. *
17 * *
18 * Warning: There are no checks to make sure *
19 * that stack limits are not exceeded. *
20 ************************************************/
21 class stack {
22         protected:
23                 int count; // Number of items in the stack
24                 int *data; // The stack data
25         public:
26                 // Initialize the stack
27                 stack(void): count(0)
28                 {
29                         data = new int[STACK_MAX];
30                 }
31                 // Destructor
32                 virtual ~stack(void) {
33                         delete data;
34                         data = NULL;
35                 }
36         private:
37                 // No copy constructor
38                 stack(const stack &);
39
40                 // No assignment operator
41                 stack & operator = (const stack &);
42         public:
43                 // Push an item on the stack
44                 void push(
45                         const int item // Item to push
46                 ) {
47                         data[count] = item;
48                         ++count;
49                 }
50                 // Remove the an item from the stack
51                 int pop(void) {
52                         --count;
```

[1] Stacked Wrong

```
53                              return (data[count]);
54                      }
55
56                      // Function to count things in
57                      // an array of stacks
58                      friend void stack_counter(
59                              stack stack_array[],
60                              const int n_stacks
61                      );
62 };
63
64 /************************************************
65 * safe_stack -- Like stack, but checks for *
66 * errors. *
67 * *
68 * Member functions: push and pop *
69 * (just like stack) *
70 ************************************************/
71 class safe_stack : public stack {
72         public:
73                 const int max; // Limit of the stack
74         public:
75                 safe_stack(void): max(STACK_MAX) {};
76                 // Destructor defaults
77         private:
78                 // No copy constructor
79                 safe_stack(const safe_stack &);
80
81                 // No assignment operator
82                 safe_stack & operator =
83                         (const safe_stack &);
84         public:
85                 // Push an item on the stack
86                 void push(
87                         // Data to push on the stack
88                         const int data
89                 ) {
90                         if (count >= (STACK_MAX-1)) {
91                                 std::cout << "Stack push error\n";
92                                 exit (8);
93                         }
94                         stack::push(data);
95                 }
96                 // Pop an item off the stack
97                 int pop(void) {
98                         if (count <= o) {
99                                 std::cout << "Stack pop error\n";
100                                exit (8);
101                        }
102                        return (stack::pop());
103                }
104 };
105
106
107 /************************************************
108 * stack_counter -- Display the count of the *
109 * number of items in an array of stacks. *
110 ************************************************/
111 void stack_counter(
112         // Array of stacks to check
113         stack *stack_array,
114
```

```
115       // Number of stacks to check
116       const int n_stacks
117 )
118 {
119       int i;
120
121       for (i = 0; i < n_stacks; ++i)
122       {
123                 std::cout << "Stack " << i << " has " <<
124                 stack_array[i].count << " elements\n";
125       }
126 }
127
128 // A set of very safe stacks for testing
129 static safe_stack stack_array[5];
130
131 int main()
132 {
133
134       stack_array[0].push(0);
135
136       stack_array[1].push(0);
137       stack_array[1].push(1);
138
139       stack_array[2].push(0);
140       stack_array[2].push(1);
141       stack_array[2].push(2);
142
143       stack_array[3].push(0);
144       stack_array[3].push(1);
145       stack_array[3].push(2);
146       stack_array[3].push(3);
147
148       stack_array[4].push(0);
149       stack_array[4].push(1);
150       stack_array[4].push(2);
151       stack_array[4].push(3);
152       stack_array[4].push(4);
153
154       stack_counter(stack_array, 5);
155       return (0);
156 }
```

(راهنمایی ۲۹۶، جواب ۷۲)

مشکلی وجود ندارد که با کودنی و جهل کافی نتوان آن را حل کرد.

برنامه ۹۲: اسم – بازی[1]

برنامه زیر چه چیزی را چاپ می کند؟

File: first.cpp
```
1 #include <string>
2
3 // The first name of the key person
4 std::string first_name = "Bill";
```

File: last.cpp
```
1 /************************************************
2 * print_name -- Print the name of a person. *
3 ************************************************/
4 #include <iostream>
5 #include <string>
6
7 // The first name
8 extern std::string first_name;
9
10 // The last name
11 std::string last_name = "Jones";
12
13 // The full name
14 std::string full_name =
15 first_name + " " + last_name;
16
17 int main()
18 {
19         // Print the name
20         std::cout << full_name << std::endl;
21         return (0);
22 }
```

(راهنمایی ۲۴۴، جواب ۳)

بعد از چند رقم اعشاری، دیگر کسی توجه نمی کند.

[1] Name – Game

برنامه ۹۳: جادو نیست[1]

چیز عجیبی در مورد کلاس info در حال رخ دادن بود. نویسنده شجاع شما موظف بود بفهمد که چه خبر است. بعد از کمی بالا و پایین کردن، من متقاعد شدم که مشکل احتمالی این است که کسی از یک اشاره گر بد استفاده کرده است و اثر آن در کل کلاس مانده است.

برای یافتن محل مشکل، من تعدادی عدد جادویی در ابتدا و انتهای داده کلاس قرار دادم. من توقع داشتم که وقتی اشتباهی رخ بدهد، این اعداد جادویی تغییر کنند. ولی بسیار شگفت زده شدم وقتی دریافتم که اشتباهات زودتر از آنچه که من انتظار داشتم اتفاق می افتند.

خب، چرا جادو از کلاس رخت برمی بندد؟

```
1 #include <stdlib.h>
2 #include <iostream>
3 #include <cstring>
4
5 /************************************************
6 * info -- A class to hold information. *
7 * *
8 * Note: *
9 * Because someone is walking all over our *
10 * memory and destroying our data, we *
11 * have put two guards at the beginning *
12 * and end of our class. If someone *
13 * messes with us these numbers will *
14 * be destroyed. *
15 * *
16 * Member functions: *
17 * set_data -- Store a string in our data. *
18 * get_data -- Get the data string. *
19 * check_magic -- Check the magic numbers. *
20 ************************************************/
21 // Magic numbers for the start and end of the
22 // data in the class info
23 const int START_MAGIC = 0x11223344;
24 const int END_MAGIC = 0x5567788;
25 class info
26 {
27         private:
28                 // Magic protection constant
29                 const int start_magic;
30
31                 // String to be stored
32                 char data[30];
33
34                 // Magic protection constant
35                 const int end_magic;
36         public:
37                 info(void):
38                         start_magic(START_MAGIC),
39                         end_magic(END_MAGIC)
40                 {}
41
42                 // Copy constructor defaults
43                 // Assignment operator defaults
44                 // Destructor defaults
```

[1] No Magic

```
45
46                  // Store some data in the class
47                  void set_data(
48                          // Data to be stored
49                          const char what[]
50                  )
51                  {
52                          strcpy(data, what);
53                  }
54
55                  // Get the data from the class
56                  char *get_data(void)
57                  {
58                          return (data);
59                  }
60
61                  // Verify that the magic
62                  // numbers are correct
63                  void check_magic(void)
64                  {
65                          if ((start_magic != START_MAGIC) ||
66                                  (end_magic != END_MAGIC))
67                          {
68                                  std::cout <<
69                                  "Info has lost its magic\n";
70                          }
71                  }
72 };
73
74 /************************************************
75 * new_info -- Create a new version of the *
76 * info class. *
77 ************************************************/
78 struct info *new_info(void)
79 {
80         struct info *result; // Newly created result.
81
82         result = (struct info *)
83         malloc(sizeof(struct info));
84
85         // Make sure the structure is clear
86         memset(result, '\0', sizeof(result));
87
88         return (result);
89 }
90 int main()
91 {
92         // An info class to play with
93         class info *a_info = new_info();
94
95         a_info->set_data("Data");
96         a_info->check_magic();
97         return (0);
98 }
```

(راهنمایی ۱۵۳، جواب ۹۸)

ناسزا، تنها زبانی است که همه برنامه نویسان آنرا درک می کنند.

برنامه ۹۴: قتل های سریع[1]

فراخوانی توابع new و delete هزینه بر هستند. اگر بخواهید برنامه خود را تسریع ببخشید و بدانید که چه کار دارید می کنید می توانید آنها را بازنویسی[2] کنید و new و delete مخصوص کلاس خود را ایجاد نمایید. این کاری است که برنامه نویس انجام داده است. الگوریتم تخصیص، بسیار ساده است، ولی بطریقی حافظه خراب می شود. چرا؟

```
1 /***********************************************
2 * bit_test -- Test out our new high speed *
3 * bit_array. *
4 ***********************************************/
5 #include <iostream>
6 #include <memory.h>
7
8 // The size of a fast bit_array.
9 // (Private to fast bit array)
10 const int BIT_ARRAY_MAX = 64; // Size in bits
11
12 // Number of bits in a byte
13 const int BITS_PER_BYTE = 8;
14 /***************************************************
15 * fast_bit_array -- A bit array using fast *
16 * allocate technology. *
17 * *
18 * Member functions: *
19 * get -- Get an element from the *
20 * array. *
21 * set -- Set the value of an element *
22 * in the array. *
23 * *
24 * new -- used to quickly allocate a bit *
25 * array. *
26 * delete -- used to quickly deallocate *
27 * a bit array. *
28 ***********************************************/
29 class fast_bit_array
30 {
31 protected:
32 // Array data
33 unsigned char
34 data[BIT_ARRAY_MAX/BITS_PER_BYTE];
35
36 public:
37 fast_bit_array(void)
38 {
39 memset(data, '\0', sizeof(data));
40 }
41 // Destructor defaults
42 private:
43 // No copy constructor
44 fast_bit_array(const fast_bit_array &);
45
46 // No assignment operator
47 fast_bit_array & operator =
48 (const fast_bit_array &);
49 public:
```

[1] Speed Kills
[2] Override

```
50                  // Set the value on an item
51                  void set(
52                          // Index into the array
53                          const unsigned int index,
54
55                          // Value to put in the array
56                          const unsigned int value
57                  )
58                  {
59                          // Index into the bit in the byte
60                          unsigned int bit_index = index % 8;
61
62                          // Byte in the array to use
63                          unsigned int byte_index = index / 8;
64
65                          if (value)
66                          {
67                                  data[byte_index] |=
68                                          (1 << bit_index);
69                          }
70                          else
71                          {
72                                  data[byte_index] &=
73                                          ~(1 << bit_index);
74                          }
75                  }
76                  // Return the value of an element
77                  int get(unsigned int index)
78                  {
79                          // Index into the bit in the byte
80                          unsigned int bit_index = index % 8;
81                          // Byte in the array to use
82                          unsigned int byte_index = index / 8;
83
84                          return (
85                                  (data[byte_index] &
86                                  (1 << bit_index)) != o);
87                  }
88                  // Allocate a new fast_bit_array
89                  void *operator new(const size_t);
90
91                  // Delete a fast bit array.
92                  void operator delete(void *ptr);
93 };
94
95 /************************************************
96 * The following routines handle the local *
97 * new/delete for the fast_bit_array. *
98 ************************************************/
99 // Max number of fast_bit_arrays we can use at once
100 const int N_FAST_BIT_ARRAYS = 30;
101
102 // If true, the bit array slot is allocated
103 // false indicates a free slot
104 static bool
105 bit_array_used[N_FAST_BIT_ARRAYS] = {false};
106
107 // Space for our fast bit arrays.
108 static char
109 bit_array_mem[N_FAST_BIT_ARRAYS]
110 [sizeof(fast_bit_array)];
111
```

```
// Handle new for "fast_bit_array".
// (This is much quicker than the
// system version of new)
/***********************************************
* fast_bit_array -- new *
* *
* This is a high speed allocation routine for *
* the fast_bit_array class. The method used *
* for this is simple, but we know that only *
* a few bit_arrays will be allocated. *
* *
* Returns a pointer to the new memory. *
***********************************************/
void *fast_bit_array::operator new(const size_t)
{
    int i; // Index into the bit array slots

    // Look for a free slot
    for (i = 0; i < N_FAST_BIT_ARRAYS; ++i)
    {
        if (!bit_array_used[i])
        {
            // Free slot found, allocate the space
            bit_array_used[i] = true;
            return(bit_array_mem[i]);
        }
    }
    std::cout << "Error: Out of local memory\n";
    exit (8);
}

/***********************************************
* fast_bit_array -- delete *
* *
* Quickly free the space used by a *
* fast bit array. *
***********************************************/
void fast_bit_array::operator delete(
    void *ptr // Pointer to the space to return
)
{
    int i; // Slot index

    for (i = 0; i < N_FAST_BIT_ARRAYS; ++i)
    {
        // Is this the right slot
        if (ptr == bit_array_mem[i])
        {
            // Right slot, free it
            bit_array_used[i] = false;
            return;
        }
    }
    std::cout <<
        "Error: Freed memory we didn't have\n";
    exit (8);
}


/***********************************************
* safe_bit_array -- A safer bit array. *
* *
```

```
174 * Like bit array, but with error checking. *
175 ************************************************/
176 class safe_bit_array : public fast_bit_array
177 {
178         public:
179                 // Sequence number generator
180                 static int bit_array_counter;
181
182                 // Our bit array number
183                 int sequence;
184
185                 safe_bit_array(void)
186                 {
187                         sequence = bit_array_counter;
188                         ++bit_array_counter;
189                 };
190                 // Destructor defaults
191         private:
192                 // No copy constructor
193                 safe_bit_array(const safe_bit_array &);
194
195                 // No assignment operator
196                 safe_bit_array & operator = (
197                                 const safe_bit_array &);
198         public:
199                 // Set the value on an item
200                 void set(
201                         // Where to put the item
202                         const unsigned int index,
203                         // Item to put
204                         const unsigned int value
205                 )
206                 {
207                         if (index >= (BIT_ARRAY_MAX-1))
208                         {
209                                 std::cout <<
210                                         "Bit array set error "
211                                         "for bit array #"
212                                         << sequence << "\n";
213                                 exit (8);
214                         }
215                         fast_bit_array::set(index, value);
216                 }
217                 // Return the value of an element
218                 int get(unsigned int index)
219                 {
220                         if (index >= (BIT_ARRAY_MAX-1))
221                         {
222                                 std::cout <<
223                                         "Bit array get error "
224                                         "for bit array #"
225                                         << sequence << "\n";
226                                 exit (8);
227                         }
228                         return (fast_bit_array::get(index));
229                 }
230 };
231
232 // Sequence information
233 int safe_bit_array::bit_array_counter = 0;
234
235 int main()
```

```
236 {
237       // Create a nice new safe bit array
238       safe_bit_array *a_bit_array =
239       new safe_bit_array;
240
241       a_bit_array->set(5, 1);
242       // Return the bit_array to the system
243       delete a_bit_array;
244       return (0);
245 }
```

(راهنمایی ۳۰۵، جواب ۵۶)

یک تکنولوژی سطح بالا از جادو قابل تمییز نیست.
آرتور س. کلارک

برنامه ۹۵: ارسال پیغام اشتباه[1]

چرا برنامه زیر، نتایج اشتباهی را تولید می کند؟

```
1 /************************************************
2 * hello -- write hello using our message system*
3 * to the log file and the screen. *
4 ************************************************/
5 #include <iostream>
6 #include <fstream>
7
8 // The log file
9 std::ofstream log_file("prog.log");
10
11 /************************************************
12 * print_msg_one -- Write a message to the *
13 * given file. *
14 ************************************************/
15 void print_msg_one(
16         // File to write the message to
17         std::ostream out_file,
18
19         // Where to send it
20         const char msg[]
21 ) {
22         out_file << msg << std::endl;
23 }
24 /************************************************
25 * print_msg -- send a message to the console *
26 * and to the log file. *
27 ************************************************/
28 void print_msg(
29         const char msg[] // Message to log
30 ) {
31         print_msg_one(std::cout, msg);
32         print_msg_one(log_file, msg);
33 }
34 int main()
35 {
36         print_msg("Hello World!");
37         return (0);
38 }
```

(راهنمایی ۳۲۸، جواب ۴۰)

[1] Sending the Wrong Message

برنامه ۹۶: تفریح ناب[1]

برنامه زیر، ایده ساده ای دارد. کلاس list، لیست پیوندی را کنترل می کند و کلاس های مشتق شده، داده ها را کنترل می نمایند. ولی وقتی اجرا می شود، به بن بست می رسد. چرا؟

```
1 /*************************************************
2 * simple linked list test. *
3 *************************************************/
4 #include <iostream>
5 #include <malloc.h>
6 #include <string>
7 /*************************************************
8 * list -- Linked list class. *
9 * Stores a pointer to void so you can *
10 * stick any data you want to in it. *
11 * *
12 * Member functions: *
13 * clear -- clear the list *
14 * add_node -- Add an item to the list *
15 *************************************************/
16 class list {
17         private:
18                 /*
19                 * Node -- A node in the linked list
20                 */
21                 class node {
22                         private:
23                                 // Data for this node
24                                 void *data;
25
26                                 // Pointer to next node
27                                 class node *next;
28
29                                 // List class does the work
30                                 friend class list;
31                                 // Constructor defaults
32                                 // Destructor defaults
33
34                                 // No copy constructor
35                                 node(const node &);
36
37                                 // No assignment operator
38                                 node &operator = (const node &);
39                         public:
40                                 node(void) :
41                                 data(NULL), next(NULL) {}
42                 };
43 //---------------------------------------
44
45                 node *first; // First node in the list
46
47                 /*
48                 * Delete the data for the node.
49                 * Because we don't know what type of
50                 * data we have, the derived class does
51                 * the work of deleting the data
52                 * through the delete_data function.
53                 */
```

[1] Pure Fun

```
54                  virtual void delete_data(void *data) = 0;
55          public:
56                  // Delete all the data in the list
57                  void clear(void) {
58                          while (first != NULL)
59                          {
60                                  // Pointer to the next node
61                                  class node *next;
62
63                                  next = first->next;
64                                  delete_data(first->data);
65                                  delete first;
66                                  first = next;
67                          }
68                  }
69
70                  // Constructor
71                  list(void): first(NULL) {};
72
73                  // Destructor. Delete all data
74                  virtual ~list(void) {
75                          clear();
76                  }
77
78                  // Add a node to the list
79                  void add_node(
80                          void *data // Data to be added
81                  ) {
82                          class node *new_node;
83
84                          new_node = new node;
85                          new_node->data = data;
86                          new_node->next = first;
87                          first = new_node;
88                  }
89 };
90 /************************************************
91 * string_list -- A linked list containing *
92 * strings. *
93 * *
94 * Uses the list class to provide a linked list *
95 * of strings. *
96 * *
97 * Member functions: *
98 * add_node -- Adds a node to the list. *
99 ************************************************/
100 class string_list : private list
101 {
102         private:
103                 // Delete a node
104                 void delete_data(
105                         void *data // Data to delete
106                 ) {
107                         free(data);
108                         data = NULL;
109                 }
110         public:
111                 // Add a new node to the list
112                 void add_node(
113                         // String to add
114                         const char *const data
115                 ) {
```

```
116                             list::add_node((void *)strdup(data));
117                     }
118 };
119
120 int main()
121 {
122         // List to test things with
123         string_list *the_list = new string_list;
124
125         the_list->add_node("Hello");
126         the_list->add_node("World");
127
128         delete the_list;
129         the_list = NULL;
130         return (0);
131 }
```

(راهنمایی ۱۱۹، جواب ۱۰۱)

فصل ۸: سردرگمی حرفه ای[1]

به یکی از دشوارترین قسمت های کتاب خوش آمدید. برنامه های این بخش طوری طراحی شده اند که حتی برنامه نویسان حرفه ای C یا C++ را پریشان خاطر سازند. شاید فکر کنید که همه چیز را درباره برنامه نویسی می دانید، ولی مسائل مطرح شده از سخت ترین و دشوارترین مسائل می باشند.

در این فصل فقط سه مسأله وجود دارد. اگر یکی را حل کردید می توانید خود را یک حرفه ای به حساب آورید. دو تا را حل کنید و من حیران می شود. اگر هر سه تا را حل کردید می توانید خود را یک قهرمان حساب کنید.

برنامه ۹۷: دوباره سلام[2]

برنامه زیر چه چیزی را چاپ می کند؟

```
1 /************************************************
2 * Normally I would put in a comment explaining *
3 * what this program is nominally used for. *
4 * But in this case I can figure out no *
5 * practical use for this program. *
6 ************************************************/
7 #include <stdio.h>
8 #include <unistd.h>
9 #include <stdlib.h>
10
11 int main()
12 {
13         printf("Hello ");
14         fork();
15         printf("\n");
16         exit(0);
17 }
```

(راهنمایی ۲۱۴، جواب ۵۰)

شکسپیر، سوالی قدیمی پرسیده است: ”To be or not to be?“ علوم کامپیوتر جواب آنرا به ما داده است:
0x2B | ~0x2B = 0xFF

وقتی این طنز را به افراد غیرفنی می گویم با تعجب به من می نگرند. افراد فنی، دقیقه ای فکر می کنند و سپی می گویند: "حق با توست". فقط یک نفر از صد نفر واقعا می خندد.

[1] Expert Confusion
[2] Hello Again

برنامه ۹۸: ضد دیباگ[1]

برنامه نویس، ایده زیرکانه ای داشت. او یک دسته کد را درون یک عبارت if(debugging) قرار داد. او سپس برنامه را اجرا کرد و وقتی خروجی دیباگ را می خواست، از دیباگر تعاملی برای تغییر debugging از ۰ به ۱ استفاده کرد. ولی این برنامه او را شگفت زده ساخت.

```
1 /***********************************************
2 * Code fragment to demonstrate how to use the *
3 * debugger to turn on debugging. All you *
4 * have to do is put a breakpoint on the "if" *
5 * line and change the debugging variable. *
6 ***********************************************/
7 extern void dump_variables(void);
8
9 void do_work()
10 {
11        static int debugging = 0;
12
13        if (debugging)
14        {
15                dump_variables();
16        }
17        // Do real work
18 }
```

(راهنمایی ۱۴۷، جواب ۸۴)

ایجاد فایل در سیستم عامل یونیکس آسان است. لذا کاربران تمایل دارند تعداد زیادی فایل را با استفاده از فضای زیاد فایل ها بسازند. گفته می شد که تنها چیز استاندارد در همه سیستم های یونیکس، پیغامی است که به کاربران می گوید فایل های خود را پاک کنند.

راهنمای اولیه مدیریت یونیکس

[1] Debug Resistant

برنامه ۹۹: فایل شبح[1]

هیچ فایلی به اسم delete.me در دایرکتوری ما وجود ندارد. پس چرا این برنامه به ما می گوید که آنرا پاک کنیم؟

```
1 /*************************************************
2 * delete_check -- Check to see if the file *
3 * delete.me exists and tell the user *
4 * to delete it if it does. *
5 *************************************************/
6 #include <iostream>
7 #include <unistd.h>
8 #include <cstdio>
9
10 int main()
11 {
12         // Test for the existence of the file
13         if (access("delete.me", F_OK)) {
14                 bool remove = true;
15         }
16         if (remove) {
17                 std::cout <<
18                         "Please remove 'delete.me'\n";
19         }
20         return (0);
21 }
```

(راهنمایی ۹۸، جواب ۳۵)

[1] Phantom File

فصل ۹: سفر به جهنم[1]

C++ بعنوان یک زبان قابل حمل فرض می شود. این عبارت "فرض می شود" خیلی دوست داشتنی است. بخاطر آن است که ما توانستیم تمام برنامه های این فصل را مهیا کنیم.

برنامه ۱۰۰: هبوط به ریو[2]

"ریو" یک پخش کننده موسیقی MP3 است. من روی چند نرم افزار لینوکس برای این دستگاه کار می کردم. هر بلوک داده با یک ساختارکنترلی ۱۶ بایتی پایان می یابد. من به دقت از عبارت struct استفاده کردم تا مطمئن شوم که ساختار بلوک درست است ولی وقتی برنامه را تست کردم، "ریو" ی من بلوک ها را از دست می داد.

خب، جریان از چه قرار است؟

```
1 /*************************************************
2 * A small part of a set of routines to *
3 * download music to a RIO mp3 player. *
4 * *
5 * Full sources for the original can be found *
6 * at http://www.oualline.com. *
7 * *
8 * This just tests the writing of the end of *
9 * block structure to the device. *
10 *************************************************/
11
12 #include <stdio.h>
13 /*
14 * The 16 byte end of block structure for a Rio.
15 * (We'd label the fields if we knew what they
16 * were.)
17 */
18 struct end_block_struct
19 {
20         unsigned long int next_512_pos; // [0123]
21         unsigned char next_8k_pos1; // [4]
22         unsigned char next_8k_pos2; // [5]
23
24         unsigned long int prev_251_pos; // [6789]
25         unsigned char prev_8k_pos1; // [10]
26         unsigned char prev_8k_pos2; // [11]
27
28         unsigned short check_sum; // [12,13]
29         unsigned short prev_32K_pos; // [14,15]
30 };
31
32 /*
33 * Macro to print offset of the
34 * field in the structure
35 */
36 #define OFFSET(what) \
37 printf(#what " %d\n", int(&ptr->what));
38
39 int main()
40 {
41         // A structure for debugging the structure
```

[1] Portage to Hell
[2] Going Down to Rio

```
42      struct end_block_struct *ptr = NULL;
43
44      printf("Structure size %d\n",
45      sizeof(end_block_struct));
46      OFFSET(next_512_pos);
47      OFFSET(next_8k_pos1);
48      OFFSET(next_8k_pos2);
49
50      OFFSET(prev_251_pos);
51      OFFSET(prev_8k_pos1);
52      OFFSET(prev_8k_pos2);
53
54      OFFSET(check_sum);
55      OFFSET(prev_32K_pos);
56      return (0);
57 }
```

(راهنمایی ۳۴۳، جواب ۱۰۳)

برنامه ۱۰۱: نقطه بدون بازگشت[1]

چرا برنامه زیر یک فایل درست را در یونیکس می نویسد و یک فایل اشتباه را در ویندوز؟ برنامه، ۱۲۸ کاراکتر را می نویسد ولی نسخه ویندوز شامل ۱۲۹ بایت است. چرا؟

```
1 /*************************************************
2 * Create a test file containing binary data. *
3 *************************************************/
4 #include <iostream>
5 #include <fstream>
6 #include <stdlib.h>
7
8 int main()
9 {
10        // current character to write
11        unsigned char cur_char;
12
13        // output file
14        std::ofstream out_file;
15
16        out_file.open("test.out", std::ios::out);
17        if (out_file.bad())
18        {
19                  std::cerr << "Can not open output file\n";
20                  exit (8);
21        }
22
23        for (cur_char = 0;
24                  cur_char < 128;
25                  ++cur_char)
26        {
27                  out_file << cur_char;
28        }
29        return (0);
30 }
```

(راهنمایی ۳۴۹، جواب ۵)

انسان جایزالخطاست. برای اینکه واقعا کارها را خراب کنید به یک کامپیوتر نیاز دارید. برای اینکه به خراب کردن ادامه دهید به بوروکراسی احتیاج دارید.

[1] Point of No Return

برنامه ۱۰۲: اعمال نیرو[1]

روی بسیاری از سیستم های یونیکس، این برنامه کار می کند ولی روی MS – DOS کار نمی کند. چرا؟

```
1 /*******************************************
2 * Check a couple of zip codes. *
3 *******************************************/
4 #include <iostream>
5
6 int main()
7 {
8         // A couple of zip codes
9         const int cleveland_zip = 44101;
10        const int pittsburgh_zip = 15201;
11
12        if (cleveland_zip < pittsburgh_zip)
13        {
14                std::cout <<
15                        "Cleveland < Pittsburgh (Wrong)\n";
16        }
17        else
18        {
19                std::cout <<
20                        "Pittsburgh < Cleveland (Right)\n";
21        }
22
23        return (0);
24 }
```

(راهنمایی ۱۰۴، جواب ۱۰۴)

[1] Zipping Along

فصل ۱۰: تعداد کمی برنامه که کار می کنند[1]

برنامه نویسان عاشق ترفند هستند. در این فصل نگاهی می اندازیم به چند برنامه که کار می کنند و از ترفندهای بسیار زیرکانه ای برای انجام کار استفاده می نمایند.

برنامه ۱۰۳: تغییر سریع[2]

سریعترین راه انجام کار زیر چیست؟

متغیر i مقدار ۱ یا ۲ دارد. اگر i، ۲ است آنرا به ۱ تغییربده. اگر i، ۱ است آنرا به ۲ تغییر بده.

(راهنمایی ۱۳۴، جواب ۴۸)

مسابقه ای به نام مسابقه C مبهم هرساله برگزار می گردد. شرکت کنندگان سعی می کنند تا راه نوشتن سخت ترین و دشوار – فهم ترین برنامه را بیابند. هر چه باشد آنها برنامه نویسند و با برنامه های دشوار – فهمی در بهترین شرایط آشنا هستند. در این مسابقه آنها باید یک برنامه را تحت بدترین شرایط بفهمند.

برخی از جوایز، عناوین جالب توجهی دارند:

- بهترین کار ساده که به روشی پیچیده انجام شده است.
- بهترین کار غیرساده که به روشی پیچیده انجام شده است.
- نامفهوم ترین برنامه.
- خوش ترکیب ترین برنامه گیج کننده.
- بیشترین شباهت برنامه با تایپ کردن تصادفی.
- کمترین تخطی از قوانین.
- نامتجانس ترین آرایش برنامه.
- بهترین تخطی از ANSI C.

[1] A Few Working Program
[2] Quick Change

برنامه ۱۰۴: نه خوب نه بد[1]

هدف عبارت if در زیربرنامه زیر چیست؟ به نظر می رسد که کاملا بی مورد باشد.

```
1 /*************************************************
2 * sum_file -- Sum the first 1000 integers in *
3 * a file. *
4 *************************************************/
5 #include <iostream>
6 #include <fstream>
7 /*************************************************
8 * get_data -- Get an integer from a file. *
9 * *
10 * Returns: The integer gotten from the file *
12 *************************************************/
12 int get_data(
13         // The file containing the input
14         std::istream &in_file
15 ) {
16         int data; // The data we just read
17         static volatile int seq = 0; // Data sequence number
19
19         ++seq;
20         if (seq == 500)
21                 seq = seq; // What's this for?
22
23         in_file.read(&data, sizeof(data));
24         return (data);
25 }
26
27 int main() {
28         int i; // Data index
29         int sum = 0; // Sum of the data so far
30
31         // The input file
32         std::ifstream in_file("file.in");
33
34         for (i = 0; i < 1000; ++i) {
35                 sum = sum + get_data(in_file);
36         }
37         std::cout << "Sum is " << sum << '\n';
38         return (0);
39 }
```

(راهنمایی ۱۷۵، جواب ۸۱)

[1] Nothing Special

برنامه ۱۰۵: تسلیم[1]

یکی از مشکلات ترفندهای جذاب این است که به ندرت برنامه نویسان توضیحی در برنامه قرار می دهند که بیان کند چه چیزی در حال اتفاق است. اینجا کدی که من در دستور stty یونیکس دیده ام بازآفرینی شده است. این برنامه چه کار می کند؟

```
1 #include <stdio.h>
2
3 int main()
4 {
5         int flags = 0x5; // Some sample flags
6
7         printf("-parity\n" + ((flags & 0x1) != 0));
8         printf("-break\n" + ((flags & 0x2) != 0));
9         printf("-xon\n" + ((flags & 0x4) != 0));
10        printf("-rts\n" + ((flags & 0x8) != 0));
11        return (0);
12 }
```

(راهنمایی ۳۰۱، جواب ۱۰۸)

[1] Waving the Flag

فصل ۱۱: ریسمانی شده، تعبیه شده – ترسیده[1]

وقتی ناسا می خواست اولین شاتل فضایی را پرتاپ کند، فضاپیما را روی سکوی پرتاپ قرار دادند، فضانورد را در اتاقک خود نشاندند و شمارش معکوس آغاز شد. آنگاه کامپیوتر یک خرابی خود – بررسی[2] را گزارش داد. آنها سعی کردند و سعی کردند و سعی کردند تا بفهمند که چه چیزی اشتباه شده است. سرآخر مجبور شدند پرتاپ را لغو کنند. مشکل به حالت رقابتی[3] برمی گشت که با شانس ۱ به ۶۴ ممکن بود هر بار در سیستم اتفاق بیفتد.

برنامه نویسانی که با چند پردازه و سیستم های تعبیه شده سروکار دارند، مشکلات خودشان را دارند. یافتن این مشکلات معمولا خیلی سخت تر از خطاهای معمولی است، چون خطاها بطور تصادفی رخ می دهند و باگ ها می توانند در مقابل آزمون، از خود مقاومت نشان دهند. علاوه براین برنامه ای که به نظر عالی و معقول می رسد می تواند خطاهای پنهانی داشته باشد.

این فصل مختص باگ های مبهم، تصادفی و شیطان صفتی است که برنامه نویس سیستم های تعبیه شده را به ستوه می آورند.

[1] Threaded, Embedded – Dreaded
[2] Self – Check
[3] Race Condition

برنامه ۱۰۶: دورریختن آشغال[1]

ما یک پورت ورودی نگاشت شده به حافظه داریم که توسط in_port_ptr اشاره گرآن است. دستگاه می تواند تا سه کاراکتر را بافر کند. برای مقداردهی اولیه دستگاه، باید بافر را خالی کنیم و چیزهای بدردنخور قدیمی را دور بریزیم. این آنچیزی است که تابع زیر قرار است انجام دهد. ولی برخی مواقع کار نمی کند. چرا؟

```
1 /************************************************
2 * clear port -- Clear the input port. *
3 ************************************************/
4 // Input register
5 char *in_port_ptr = (char *)0xFFFFFFE0;
6
7 // Output register
8 char *out_port_ptr = (char *)0xFFFFFFE1;
9
10 /************************************************
11 * clear_input -- Clear the input device by *
12 * reading enough characters to empty the *
13 * buffer. (It doesn't matter if we read *
14 * extra, just so long as we read enough.)*
15 ************************************************/
16 void clear_input(void)
17 {
18         char ch; // Dummy character
19
20         ch = *in_port_ptr; // Grab data
21         ch = *in_port_ptr; // Grab data
22         ch = *in_port_ptr; // Grab data
23 }
```

(راهنمایی ۱۲۹، جواب ۹)

قانون اول بهینه سازی برنامه آن را انجام نده. قانون دوم بهینه سازی برنامه بازهم آن را انجام نده.

[1] Takin Out the Trash

برنامه ۱۰۷: آشغال جمع کن بهتر

مشکل برنامه ۱۰۶را با اضافه کردن کلمه کلیدی volatile حل کرده ایم. ولی باز هم کارها درست پیش نمی رود.

```
1 /*************************************************
2 * clear port -- Clear the input port. *
3 *************************************************/
4 // Input register
5 const char *volatile in_port_ptr =
6 (char *)0xFFFFFFE0;
7
8 // Output register
9 const char *volatile out_port_ptr =
10 (char *)0xFFFFFFE1;
11
12 /*************************************************
13 * clear_input -- Clear the input device by *
14 * reading enough characters to empty the *
15 * buffer. (It doesn't matter if we read *
16 * extra, just so long as we read enough.)*
17 *************************************************/
18 void clear_input(void)
19 {
20         char ch; // Dummy character
21
22         ch = *in_port_ptr; // Grab data
23         ch = *in_port_ptr; // Grab data
24         ch = *in_port_ptr; // Grab data
25 }
```

(راهنمایی ۳۳۶، جواب ۶۱)

کاربری مشکل بزرگی داشت و تقاضای پشتیبانی فنی کرد. تکنیسین ساعات ها تلاش نمود تا مشکل را تلفنی برطرف سازد ولی نتوانست لذا از کاربر خواست که یک کپی از دیسک خود را برای او بفرستد. روز بعد توسط فدرال اکسپرس، نامه ای به دست تکنیسین رسید که یک فتوکپی از دیسک در آن بود. کاربر زیاد احمق نبود. او می دانست که دیسک دو طرفه است و از هر دو طرف کپی گرفته بود.

بطور شگفت آوری، تکنیسین قادر بود از طریق فتوکپی، مشکل را پیدا کند. او فهمید که کاربر نسخه اشتباهی از نرم افزار را در اختیار دارد.

برنامه ۱۰۸: کوتاه مدت[1]

برنامه نویس نیاز داشت تا یک تأخیر کوتاه در برنامه اش ایجاد کند. او فهمید که اگر ۱۸۶۳ عمل ضرب را انجام دهد، به تأخیر مورد نظر می رسد. این حقیقت به زیربرنامه زیر منجر شد. ولی تحت بعضی شرایط، این تابع کار نمی کند. چرا؟

```
1 /************************************************
2 * bit_delay -- Delay one bit time for *
3 * serial output. *
4 * *
5 * Note: This function is highly system *
6 * dependent. If you change the *
7 * processor or clock it will go bad. *
8 ************************************************/
9 void bit_delay(void)
10 {
11        int i; // Loop counter
12        int result; // Result of the multiply
13
14        // We know that 1863 multiplies delay
15        // the proper amount
16        for (i = 0; i < 1863; ++i)
17        {
18                result = 12 * 34;
19        }
20 }
```

(راهنمایی ۳۴۲، جواب ۱۶)

[1] Short Time

برنامه ۱۰۹: کوتاه مدت، تجدیدنظرشده[1]

برنامه نویس سعی کرد تا مشکل برنامه ۱۰۸ را با تغییر عملوندهای ضرب به متغیرها حل کند. ولی حلقه هنوز هم خیلی کوتاه است. جریان از چه قرار است؟

```
1 /*************************************************
2 * bit_delay -- Delay one bit time for *
3 * serial output. *
4 * *
5 * Note: This function is highly system *
6 * dependent. If you change the *
7 * processor or clock it will go bad. *
8 *************************************************/
9 void bit_delay(void)
10 {
11        int i; // Loop counter
12        int result; // Result of the multiply
13
14        // Factors for multiplication
15        int factor1 = 12;
16        int factor2 = 34;
17
18        // We know that 1863 multiples
19        // delay the proper amount
20        for (i = 0; i < 1863; ++i)
21        {
22                result = factor1 * factor2;
23        }
24 }
```

(راهنمایی ۱۰۷، جواب ۸۹)

[1] Short Time Revisited

برنامه ۱۱۰: کوتاه مدت ۳[1]

مشکل برنامه ۱۰۹ حل شد. حالا تأخیر به مقدار مورد انتظار نزدیک تر شده است. دقیقا آنچه که می خواستیم نیست ولی نزدیک به آن است. باز چه شده است؟

```
1 /************************************************
2 * bit_delay -- Delay one bit time for *
3 * serial output, *
4 * *
5 * Note: This function is highly system *
6 * dependent. If you change the *
7 * processor or clock it will go bad. *
8 ************************************************/
9 void bit_delay(void)
10 {
11        int i; // Loop counter
12        volatile int result;// Result of the multiply
13
14        // Factors for multiplication
15        int factorl = 12;
16        int factor2 = 34;
17
18        // We know that 1863 multiplies
19        // delay the proper amount
20        for (i = 0; i < 1863; ++i)
21        {
22                result = factor1 * factor2;
23        }
24 }
```

(راهنمایی ۹۵، جواب ۳۹)

برای مقادیر بسیار بزرگ ۱، ۱ برابر ۲ می شود.

[1] Short Time III

برنامه ۱۱۱: دست اندازی در مسیر مسابقه[1]

این برنامه، دو ریسمان[2] را آغاز می کند. یکی داده را درون یک بافر می ریزد و دیگری داده را درون یک فایل می ریزد. ولی برخی اوقات، داده خراب می شود. چرا؟

```
1 /************************************************
2 * Starts two threads *
3 * *
4 * 1) Reads data from /dev/input and puts *
5 * it into a buffer. *
6 * *
7 * 2) Takes data from the buffer and *
8 * writes the data to /dev/output.*
9 ************************************************/
10 #include <cstdio>
11 #include <stdlib.h>
12 #include <pthread.h>
13 #include <unistd.h>
14 #include <sys/fcntl.h>
15
16 static const int BUF_SIZE = 1024; // Buffer size
17 static char buffer[BUF_SIZE]; // The data buffer
18
19 // Pointer to end of buffer
20 static char *end_ptr = buffer + BUF_SIZE;
21
22 // Next character read goes here
23 static char *in_ptr = buffer;
24
25 // Next character written comes from here
26 static char *out_ptr = buffer;
27
28 static int count = 0; // Number of characters in the buffer
29
30 /************************************************
31 * reader -- Read data and put it in the global *
32 * variable buffer. When data is *
33 * installed the variable count is *
34 * increment and the buffer pointer *
35 * advanced. *
36 ************************************************/
37 static void *reader(void *) {
38         // File we are reading
39         int in_fd = open("/dev/input", 0_RDONLY);
40
41         while (1) {
42                 char ch; // Character we just got
43
44                 while (count >= BUF_SIZE)
45                         sleep(1);
46
47                 read(in_fd, &ch, 1);
48
49                 ++count;
50                 *in_ptr = ch;
51                 ++in_ptr;
52
```

[1] A bumb on the Race Track
[2] Thread

```
53                      if (in_ptr == end_ptr)
54                      in_ptr = buffer;
55              }
56 }
57
58 /*********************************************
59 * writer -- Write data from the buffer to *
60 * the output device. Gets the data *
61 * from the global buffer. Global variable*
62 * count is decrement for each character *
63 * taken from the buffer and the buffer *
64 * pointer advanced. *
65 *********************************************/
66 static void writer(void)
67 {
68              // Device to write to
69              int out_fd = open("/dev/output", 0_RDONLY);
70
71              while (1) {
72                      char ch; // Character to transfer
73
74                      while (count <= 0)
75                              sleep(1);
76
77                      ch = *out_ptr;
78
79                      --count;
80                      ++out_ptr;
81
82                      if (out_ptr == end_ptr)
83                              out_ptr = buffer;
84
85                      write(out_fd, &ch, 1);
86              }
87 }
88
89 int main() {
90              int status; /* Status of last system call */
91
92              /* Information on the status thread */
93              pthread_t reader_thread;
94
95              status = pthread_create(&reader_hread, NULL, reader, NULL);
96
97              if (status != 0) {
98                      perror("ERROR: Thread create failed:\n ");
99                      exit (8);
100             }
101
102             writer();
103             return (0);
104 }
```

(راهنمایی ۲۲۲، جواب ۹۲)

در طی سالیان، نصب کننده های سیستم راه های گوناگونی را برای کابل کشی سقف های کاذب ایجاد کرده اند. یکی از مبتکرانه ترین آنها، روش "سگ کوچک" است. یک نفر سگ کوچکی را می گیرد، رشته ای را به گردنش می بندد و سگ را درون سقف قرار می دهد. سپس صاحب به سمت سوراخی می رود که می خواهند کابل از آنجا بیرون بیاید و سگ را صدا می زند. سگ به طرف صاحبش می دود. آنها یک کابل به رشته می بندند و از این طریق، کابل کشی را انجام می دهند.

برنامه ۱۱۲: زودباش و صبرکن[1]

بنا به دلایلی این برنامه برای مدتی کار می کند و سپس متوقف می شود.

```
1 #include <cstdio>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <sys/fcntl.h>
5
6 // Resource protection mutexes
7 static pthread_mutex_t resource1 =
8 PTHREAD_MUTEX_INITIALIZER;
9
10 static pthread_mutex_t resource2 =
11 PTHREAD_MUTEX_INITIALIZER;
12
13 /************************************************
14 * A couple of routines to do work. Or they *
15 * would do work if we had any to do. *
16 ************************************************/
17 static void wait_for_work(void) {} // Dummy
18 static void do_work(void) {} // Dummy
19
20 /************************************************
21 * process_1 -- First process of two. *
22 * *
23 * Grab both resources and then do the work *
24 ************************************************/
25 static void *process_1(void *)
26 {
27         while (1) {
28                 wait_for_work();
29
30                 pthread_mutex_lock(&resource1);
31                 pthread_mutex_lock(&resource2);
32
33                 do_work();
34
35                 pthread_mutex_unlock(&resource2);
36                 pthread_mutex_unlock(&resource1);
37         }
38 }
39
40 /************************************************
41 * process_2 -- Second process of two. *
42 * *
43 * Grab both resources and then do the work. *
44 * (but slightly different work from *
45 * process_1) *
46 ************************************************/
47 static void process_2(void)
48 {
49          while (1) {
50                 wait_for_work();
51
52                 pthread_mutex_lock(&resource2);
53                 pthread_mutex_lock(&resource1);
54
```

[1] Hurry Up and Wait

```
55                      do_work();
56
57                      pthread_mutex_unlock(&resources1);
58                      pthread_mutex_unlock(&resource2);
59              }
60 }
61
62 int main()
63 {
64              int status; /* Status of last system call */
65
66              /* Information on the status thread */
67              pthread_t thread1;
68
69              status = pthread_create(&thread1,
70                                      NULL, process_1, NULL);
71
72              if (status != o) {
73                      perror(
74                                      "ERROR: Thread create failed:\n ");
75                      exit (8);
76              }
77
78              process_2();
79              return (0);
80 }
```

(راهنمایی ۹۷، جواب ۲۴)

برنامه ۱۱۳: تسلیم[1]

این برنامه، قسمت کوچکی از درایور ترمینال یونیکس را در خود دارد. (درایور ترمینال یونیکس، تعداد زیادی پرچم[2] دارد.)

وقتی این برنامه روی یک کامپیوتر Celerity C1000 قرار گرفت، مشکلات آغاز شدند. حدود یک بار در هفته، پرچم ها بطور اسرارآمیزی روشن یا خاموش می شدند. می توانید بگویید مشکل کجاست؟

```
1 /************************************************
2 * flag -- Demonstrate the use of flag setting *
3 * and clearing. This is a demonstration *
4 * program that does not run in real life. *
5 * But it is a good example of a very tiny *
6 * part of the code in a terminal driver. *
7 ************************************************/
8 #include <cstdio>
9 #include <stdlib.h>
10 #include <pthread.h>
11
12
13 const char XOFF = 'S' - '@';// Turns off output
14 const char XON = '0' - '@'; // Turns on output
15
16 static int flags = 0; // State flags
17 //
18 // ^S in effect
19 const int STOP_OUTPUT = (1 << 0);
20
21 // CD is present
22 const int CD_SIGNAL = (1 << 1);
23
24 /************************************************
25 * read_ch -- read a single character. *
26 * *
27 * Returns the character read. *
28 ************************************************/
29 static char read_ch(void)
30 {
31         // Dummy function
32         return ('x');
33 }
34
35 /************************************************
36 * write_ch -- write a character to the output *
37 * (Whatever that is.) *
38 ************************************************/
39 static void write_ch(const char ch)
40 {
41         // Dummy function
42 }
43 /************************************************
44 * do_input -- handle the reading and *
45 * processing of characters. *
46 ************************************************/
47 static void *do_input(void *)
48 {
```

[1] Flag Waving
[2] Flag

```
49          while (1)
50          {
51                      char ch; // Character we just read
52
53                      ch = read_ch();
54
55                      switch (ch) {
56                                  case XOFF:
57                                              flags |= STOP_OUTPUT;
58                                              break;
59                                  case XON:
60                                              flags &= ~STOP_OUTPUT;
61                                              break;
62                                  default:
63                                              write_ch(ch);
64                                              break;
65                      }
66          }
67 }
68
69 /************************************************
70 * wait_for_cd_change -- wait for the CD signal *
71 * to change and return the value of the *
72 * signal. *
73 ************************************************/
74 static int wait_for_cd_change(void)
75 {
76          // Dummy
77          return (1);
78 }
79 /************************************************
80 * do_signals -- Monitor signals and set flags *
81 * based on the signal changes. *
82 ************************************************/
83 void do_signals(void)
84 {
85          while (1) {
86                      // The current cd level
87                      int level = wait_for_cd_change();
88                       if (level) {
89                                  flags |= CD_SIGNAL;
90                      } else {
91                                  flags &= ~CD_SIGNAL;
92                      }
93          }
94 }
95
96 int main()
97 {
98          int status; // Status of last system call
99
100         // Information on the status thread
101         pthread_t input_thread;
102
103         status = pthread_create(&input_thread,
104                                 NULL, do_input, NULL);
105
106         if (status != 0) {
107                     perror(
108                                 "ERROR: Thread create failed:\n ");
109                     exit (8);
110         }
```

```
111
112         do_signals();
113         return(o);
114 }
```

(راهنمایی ۲۲، جواب ۵۲)

ریشه اصلی مشکلات، راه حل ها هستند.

برنامه ۱۱۴: پیشرفت کند[1]

این برنامه از دو ریسمان تشکیل شده است. اولی، sum، کاری انجام می دهد که زمان زیادی می برد. دومی، status_monitor، هر بار که کاربر، کلید بازگشت را فشار می دهد، یک گزارش پیشرفت می دهد. ولی بعد از تعدادی اجرای آزمایشی، برنامه نویس به درستی گزارش های پیشرفت شک کرد. چرا؟

```
1 /*************************************************
2 * Sum -- This program sums the sine of the *
3 * numbers from 1 to MAX. (For no good *
4 * reason other than to have something *
5 * to do that takes a long time.) *
6 * *
7 * Since this takes a long time, we have a *
8 * second thread that displays the progress of *
9 * the call. *
10 *************************************************/
11 #include <cstdio>
12 #include <cmath>
13 #include <pthread.h>
14 #include <stdlib.h>
15
16 /* Counter of what we've summed so far */
17 static int counter;
18
19 /*************************************************
20 * status_monitor -- Monitor the status and *
21 * tell the user how far things have *
22 * progressed. *
23 * *
24 * This thread merely waits for the user to *
25 * press <enter> and then reports the current *
26 * value of counter. *
27 *************************************************/
28 static void *status_monitor(void *) {
29         /* buffer to stuff that comes in */
30         char buffer[3];
31
32         while (1) {
33                 fgets(buffer, sizeof(buffer), stdin);
34                 printf("Progress %d\n", counter);
35                 fflush(stdout);
36         }
37 }
38
39 /*************************************************
40 * sum -- Sum the sine of the numbers from 0 to *
41 * 0x3FFFFFFF. Actually we don't care *
42 * about the answer, all we're trying to *
43 * do is create some sort of compute *
44 * bound job so that the status_monitor *
45 * can be demonstrated. *
46 *************************************************/
47 static void sum(void) {
48         static double sum = 0; /* Sum so far */
49
50         for (counter = 0;
51                 counter < 0x3FFFFFF;
52                 ++counter)
```

[1] Slow Progress

```
53          {
54                  sum += sin(double(counter));
55          }
56
57          printf("Total %f\n", sum);
58          exit (0);
59 }
60
61 int main() {
62          // Status of last system call
63          int status;
64
65          // Information on the status thread
66          pthread_t status_thread;
67
68          status = pthread_create(&status_thread, NULL,
69          status_monitor, NULL);
70
71          if (status != o) {
72                  perror(
73                          "ERROR: Thread create failed:\n ");
74                  exit (8);
75          }
76
77          sum();
78
79          return(0);
80 }
```

(راهنمایی ۳۵۰، جواب ۱۱۴)

قسمت دوم: راهنمایی ها

راهنمایی ۱: در روزهای نخستین راه آهن، وقتی خطوط راه آهن همدیگر را قطع می کردند، در برخی مواقع، قطارها به یکدیگر برخورد می کردند. بنابراین، قانونی وضع شد:

وقتی دو قطار در محل تقاطع خطوط به هم نزدیک می شوند، هر دو باید توقف کنند و متوقف بمانند تا زمانیکه آن یکی رد شود.

(جواب ۲۴)

راهنمایی ۲: یونیکس از <line-feed> برای انتهای خطوط استفاده می کند. ویندوز مایکروسافت از <carriage-return><line-feed> استفاده می کند. (جواب ۵)

راهنمایی ۳: عبارت if (n2 =! 0) بغیر از گیج کردن شما، کاری نمی کند که در این حالت، کار خود را به نحو احسن انجام می دهد. (بعدی راهنمایی ۸۲، جواب ۲۵)

راهنمایی ۴: تابع سازنده کار مقداردهی اولیه اعداد جادویی را بخوبی انجام می دهد. یا اگر فراخوانی شود به خوبی انجام می دهد. ولی همه می دانند که بدون فراخوانی تابع سازنده نمی توان متغیری ایجاد نمود. (بعدی راهنمایی ۳۰۰، جواب ۹۸)

راهنمایی ۵: تفاوت بین ماکروهای ساده و ماکروهای پارامتردار چیست؟ (جواب ۱۱۳)

راهنمایی ۶: فراخوانی های سیستم عامل، هزینه بر هستند. (جواب ۹۶)

راهنمایی ۷: unsigned char (بعدی راهنمایی ۳۱۳، جواب ۱۱)

راهنمایی ۸: قانون قطع درخت: شاخه ای را که روی آن ایستاده اید، قطع نکنید. (بعدی راهنمایی ۳۱۷، جواب ۷۵)

راهنمایی ۹: برنامه با ماشین هایی با مجموعه دستورالعمل های پیچیده مثل پردازنده های 80x86 مشکلی ندارد ولی روی ماشین های RISC مثل Sparc مشکل دارد. همچنین روی Celerity 1000 که من روی آن مشکل پیدا کردم، درست کار نمی کند. (بعدی راهنمایی ۱۴۳، جواب ۵۲)

راهنمایی ۱۰: این برنامه در هر بار، یک کاراکتر را می خواند. انتظار می رود که در هر بار یک کاراکتر را نیز بنویسد. (بعدی راهنمایی ۱۰۲، جواب ۹۹)

راهنمایی ۱۱: وقتی برنامه نویس سعی می کند که متغیر دیباگینگ را مقداردهی کند، این پیغام خطا را دریافت می کند:

debugging – no such variable or class

(بعدی راهنمایی ۱۰۵، جواب ۸۴)

راهنمایی ۱۲: خروجی پیش پردازنده را بررسی کنید. (جواب ۸۲)

راهنمایی ۱۳: کامپایلر g++ این اخطارها را می دهد:

```
var.cpp: In function 'int main()':
var.cpp:14: warning: unused variable 'bool remove'
var.cpp:16: warning: the address of 'int remove(const char*)', will always be 'true'
```

(جواب ۳۵)

راهنمایی ۱۴: بهینه ساز[1] دارد با برنامه شما بازی می کند. (جواب ۱۱۴)

راهنمایی ۱۵: نتایج، وابسته به سیستم هستند. (بعدی راهنمایی ۲۷۸، جواب ۶۳)

راهنمایی ۱۶: M_PI صحیح است ولی نتیجه غلط چاپ می شود. (بعدی راهنمایی ۱۷۰، جواب ۱۰)

راهنمایی ۱۷: عملگر کاما، نتیجه عبارت دوم را برمی گرداند لذا عبارت 5, 9 مقدار 9 را دارد. (بعدی راهنمایی ۳۴۸، جواب ۸۶)

راهنمایی ۱۸: شما نمی توانید. (بعدی راهنمایی ۳۴۴، جواب ۸۰)

راهنمایی ۱۹: تابع printf گم می شود و شروع به ساختن نتایج غیرواقعی می کند. (بعدی راهنمایی ۳۱، جواب ۸۵)

راهنمایی ۲۰: تعداد دفعاتی که بدنه حلقه اجرا می شود، احتمالا از آنچه که فکر می کنید، کمتر است. (بعدی راهنمایی ۳۶، جواب ۸۹)

راهنمایی ۲۱: اگر از MS – DOS استفاده می کنید، نتایج به مدل حافظه بستگی دارد. (بعدی راهنمایی ۱۳۰، جواب ۲۱)

راهنمایی ۲۲: خرابی، وابسته به سیستم است. (بعدی راهنمایی ۹، جواب ۵۲)

راهنمایی ۲۳: چه وقتی مخرب[2] true_name فراخوانی می شود؟ از رشته در چه زمانی استفاده می شود؟ (جواب ۳۰)

راهنمایی ۲۴: هر چه تعداد توابعی که بین فراخوانی tmp_name و استفاده از نتایج، فراخوانی می شوند بیشتر باشد، احتمال اینکه نتایج نادرست بگیرید بیشتر است. (بعدی راهنمایی ۸۵، جواب ۱۸)

راهنمایی ۲۵: C++ فقط تا حدودی نوع – مطمئن[3] است. (بعدی راهنمایی ۶۳، جواب ۷)

راهنمایی ۲۶: داده های ایستا، خطرناک هستند. (جواب ۱۰۰)

راهنمایی ۲۷: منابع مورد نیاز resource1, resource2 هستند یا resource2, resource1؟ (بعدی راهنمایی ۱، جواب ۲۴)

راهنمایی ۲۸: برنامه را از طریق پیش پردازنده اجرا کنید. (بعدی راهنمایی ۳۲۷، جواب ۲۹)

[1] Optimizer
[2] Destructor
[3] Typesafe

راهنمایی ۲۹: چندبار حلقه اجرا می شود؟ (بعدی راهنمایی ۲۰، جواب ۸۹)

راهنمایی ۳۰: کامپایلرهای بورلند به شما این اجازه را می دهند که در زمان کامپایل مشخص کنید که پیش فرض متغیرهای کاراکتری، علامت دار یا بدون علامت است. (بعدی راهنمایی ۶۰، جواب ۸)

راهنمایی ۳۱: C در فراخوانی های printf، پارامترها را بازرسی نمی کند. (بعدی راهنمایی ۲۷۷، جواب ۸۵)

راهنمایی ۳۲: جواب به این بستگی دارد که چه کسی کتابخانه مدیریت هیپ[1] را نوشته است. (جواب ۷۷)

راهنمایی ۳۳: برنامه را از طریق پیش پردازنده اجرا کنید و نتایج را ببینید. (بعدی راهنمایی ۱۷۹، جواب ۱۰۵)

راهنمایی ۳۴: نتایج بدین صورت هستند:

```
11072   12627   16262
3157    3664    5034
13605   16307   22366
```

(بعدی راهنمایی ۱۵۸، جواب ۵۳)

راهنمایی ۳۵: سینتکس پیش پردازنده، سینتکس C++ نیست. (بعدی راهنمایی ۲۸۴، جواب ۸۲)

راهنمایی ۳۶: بعد از اینکه result را محاسبه کردیم، با آن چه می کنیم؟ (بعدی راهنمایی ۱۵۲، جواب ۸۹)

راهنمایی ۳۷: هر کسی که بصورت i++++ برنامه می نویسد باید کشته شود. (بعدی راهنمایی ۲۷۲، جواب ۸۷)

راهنمایی ۳۸: عبارت counter == 10; یک عبارت معتبر C++ است. هیچ کاری نمی کند ولی معتبر است. (بعدی راهنمایی ۲۰۵، جواب ۱۱۲)

راهنمایی ۳۹: می توانید 1/3 را دقیقا بصورت یک عدد دهدهی نشان دهید؟ آیا کامپیوتر می تواند 0.1 را دقیقا بصورت یک عدد ممیز شناور نشان دهد؟ (جواب هر دو سوال یکی است) (جواب ۱۰۷)

راهنمایی ۴۰: مشکل در یکی از خط های قبلی و نه در خط ۱۶ وجود دارد. (بعدی راهنمایی ۳۴۶، جواب ۷۹)

راهنمایی ۴۱: height هیچ وقت مقدار ۲ به خود نمی گیرد. (بعدی راهنمایی ۷۸، جواب ۶۲)

راهنمایی ۴۲: دیکشنری file به ترتیب الفبایی است. (بعدی راهنمایی ۳۱۱، جواب ۷۴)

راهنمایی ۴۳: an_array.operator = (an_array) چه کار می کند؟ (جواب ۷۵)

راهنمایی ۴۴: توگذاری[2]، اشتباه است. (بعدی راهنمایی ۱۵۶، جواب ۳۱)

راهنمایی ۴۵: کامپایلر g++ این اخطار را می دهد:

```
semi.cpp: In function 'int main()':
semi.cpp:15: warning: statement with no effect
```

(بعدی راهنمایی ۳۵، جواب ۸۲)

[1] Heap
[2] Indentation

راهنمایی ۴۶: هرچه می بینید، همان را می گیرید[1]. (بعدی راهنمایی ۳۰۷، جواب ۶۹)

راهنمایی ۴۷: خروجی این است: One million 1 (بعدی راهنمایی ۵۹، جواب ۴۴)

راهنمایی ۴۸: در پایان فراخوانی تابع، بر سر بافر چه می آید؟ (جواب ۸۳)

راهنمایی ۴۹: برای برخی عبارات چندقسمته، کامپایلر می تواند ترتیب اجرا را معین کند. (جواب ۲۶)

راهنمایی ۵۰: این برنامه هیچ اخطار کامپایلری ندارد. (بعدی راهنمایی ۳۱۸، جواب ۲۰)

راهنمایی ۵۱: در رابطه با آنچه که در داده struct قرار داده می شود، دو مشکل وجود دارد. (جواب ۷۱)

راهنمایی ۵۲: شما می دانید این راهنمایی چه باید باشد، نه؟ (بعدی راهنمایی ۲۰۷، جواب ۴۲)

راهنمایی ۵۳: دو تابع داریم که در یک بازگشت نامتناهی، هر نتیجه را فراخوانی می کنند. از آنجا که فقط سه تابع عضو داریم، یافتن اینکه کدام مشکل ایجاد می کنند، سخت نخواهد بود. (بعدی راهنمایی ۱۲۵، جواب ۱۲)

راهنمایی ۵۴: تقدم عملگرها. (جواب ۴۹)

راهنمایی ۵۵: من نمی دانستم که می شود در اعداد C++، کاما قرار داد. (بعدی راهنمایی ۳۳۵، جواب ۴۴)

راهنمایی ۵۶: delete کجا فراخوانی می شود؟ (جواب ۳۲)

راهنمایی ۵۷: دقت مضاعف، ۶۴ بیت است. استاندارد C این است که همه ممیزشناورها را بصورت double داشته باشیم. (بعدی راهنمایی ۹۴، جواب ۷۳)

راهنمایی ۵۸: 0d کد اسکی carriage return است. (بعدی راهنمایی ۲۳۴، جواب ۵)

راهنمایی ۵۹: کامپایلر g++ این اخطار را می دهد:

```
comma.cpp: In function 'int main()':
comma.cpp:12: warning: left-hand operand of comma expression has no effect
```

(بعدی راهنمایی ۱۲۶، جواب ۴۴)

راهنمایی ۶۰: کامپایلر g++ این اخطار را می دهد:

```
chff.cpp: In function `int main()':
chff.cpp:13: warning: comparison is always 0 due to limited range of data type
```

(جواب ۸)

راهنمایی ۶۱: استثناء گرفته نشده[2] مربوط به کلاس problem است. (بعدی راهنمایی ۳۳۹، جواب ۵۵)

راهنمایی ۶۲: کاراکتر "A" مقدار ۶۵ را دارد. مقدار 1 + "A" برابر ۶۶ است. این مسأله مربوط به این خروجی است:

[1] What you see is what you get
[2] Uncaught Exception

A6667

(جواب ۴۵)

راهنمایی ۶۳: extern ها در C++، نوع – امن نیستند. (جواب ۷)

راهنمایی ۶۴: فراخوانی سیستم fork یک پردازه جدا با حافظه جدا ایجاد می کند. (بعدی راهنمایی ۲۵۲، جواب ۵۰)

راهنمایی ۶۵: برنامه از اساس غلط است. (بعدی راهنمایی ۲۸۲، جواب ۱۱۵)

راهنمایی ۶۶: توگذاری، رعایت نشده است. (جواب ۹۷)

راهنمایی ۶۷: روی درست بودن توگذاری حساب نکنید. (جواب ۱۳)

راهنمایی ۶۸: کد خروجی که توسط این برنامه به سیستم عامل برگردانده می شود چیست؟ (جواب ۶)

راهنمایی ۶۹: نتایج وابسته به سیستم هستند. (بعدی راهنمایی ۲۷۹، جواب ۹۴)

راهنمایی ۷۰: چند اجرای نمونه:

```
Enter two integers: 100 3
Result is: 100

Enter two integers: 37 0
Result is: 37
```

(بعدی راهنمایی ۳، جواب ۲۵)

راهنمایی ۷۱: کاراکتر \n جائیکه نباید، ظاهر می شود. (جواب ۳۳)

راهنمایی ۷۲: رشته های C++ همه کارها را برای ما انجام می دهند. ولی یک کار را بدون اطلاع ما انجام می دهند و همان مشکل زا است. (بعدی راهنمایی ۱۶۲، جواب ۳۶)

راهنمایی ۷۳: برنامه وقتی اجرا می شود، درست کار نمی کند. (بعدی راهنمایی ۱۸۲، جواب ۹۵)

راهنمایی ۷۴: در اغلب سیستم ها، دستور $program کار می کند و دستور $program >output.txt کار نمی کند. (بعدی راهنمایی ۱۹۷، جواب ۸۳)

راهنمایی ۷۵: out_file چه نوع پارامتری است؟ (بعدی راهنمایی ۱۵۹، جواب ۴۰)

راهنمایی ۷۶: در مبنای دو، ۳ برار ۰۰۱۱ و ۱۲ برابر ۱۱۰۰ است. (بعدی راهنمایی ۲۱۸، جواب ۱۷)

راهنمایی ۷۷: وقتی open درست کار نمی کند، خطا رخ می دهد. (بعدی راهنمایی ۲۸۸، جواب ۶۰)

راهنمایی ۷۸: عبارت 11 height = 2 یک عبارت قابل اجرا نیست. (بعدی راهنمایی ۲۸۷، جواب ۶۲)

راهنمایی ۷۹: چه زمانی a_var مقداردهی اولیه شده و تابع سازنده فراخوانی می گردد؟ (بعدی راهنمایی ۱۳۷، جواب ۱۱۱)

راهنمایی ۸۰: کامپیوترها، ریاضی بلد نیستند. (بعدی راهنمایی ۲۶۸، جواب ۱)

راهنمایی ۸۱: هیچ پیش الگو[1] – هیچ بررسی پارامتر. (بعدی راهنمایی ۱۷۴، جواب ۴۱)

راهنمایی ۸۲: کامپایلر g++ این اخطار را می دهد:

```
not_z.cpp: In function `int main()':
not_z.cpp:13: warning: suggest parentheses around assignment used as truth value
```

(بعدی راهنمایی ۲۶۲، جواب ۲۵)

راهنمایی ۸۳: برای هر دو اعلان، یک متغیر تعریف شده است. (بعدی راهنمایی ۱۴۸، جواب ۵۷)

راهنمایی ۸۴: جواب 1 << 0x8000 چیست؟ (جواب ۱۹)

راهنمایی ۸۵: به چه چیزی اشاره شده است؟ صاحب آن کیست؟ برای چه مدتی؟ (جواب ۱۸)

راهنمایی ۸۶: چه کسی صاحب داده مورد اشاره از جانب اشاره گرها است؟ (بعدی راهنمایی ۲۶، جواب ۱۰۰)

راهنمایی ۸۷: نتایج وابسته به سیستم هستند. (بعدی راهنمایی ۲۱، جواب ۲۱)

راهنمایی ۸۸: جواب عبارت printf(“That’s all\n”); کجا قرار می گیرد؟ (بعدی راهنمایی ۴۸، جواب ۸۳)

راهنمایی ۸۹: این، C++ صحیح، مجاز و استاندارد است هرچند ممکن است به نظر برخی این طور نباشد. (بعدی راهنمایی ۲۱۱، جواب ۸۶)

راهنمایی ۹۰: خروجی را از طریق پیش پردازنده اجرا کنید. (بعدی راهنمایی ۲۷۳، جواب ۸۸)

راهنمایی ۹۱: کامپایلر g++ این اخطار را می دهد:

```
hbit.cpp: In function 'void bit_out(short int)':
hbit.cpp:19: warning: overflow in implicit constant conversion
```

(جواب ۲)

راهنمایی ۹۲: واضح است که مشکل باید قبل از خط ۲۸ باشد چون ما پیغام ...Starting را نمی بینیم. (بعدی راهنمایی ۱۱۱، جواب ۶۸)

راهنمایی ۹۳: خروجی این است:

```
i is 3
i is 2
```

(جواب ۸۷)

راهنمایی ۹۴: این مسأله وابسته به پیاده سازی است. در برخی سیستم های قدیمی تر، مقدار درست دقت بیت ها را دریافت می کنید. در سیستم هایی که ممیز شناور را شبیه سازی می کنند، نتایج دقیق ارائه می شود و سیستم هایی با کمک پردازنده، نتایج اغراق آمیز نشان می دهند. (جواب ۷۳)

راهنمایی ۹۵: نتایج وابسته به زمان اجرا می باشند. (بعدی راهنمایی ۳۳۱، جواب ۳۹)

[1] Prototype

راهنمایی ۹۶: اخطارهای gcc اینگونه اند:

```
sum.c: In function 'sum':
sum.c:13: warning: declaration of 'i1' shadows a parameter
sum.c:14: warning: declaration of 'i2' shadows a parameter
sum.c:15: warning: declaration of 'i3' shadows a parameter
```

(جواب ۹۴)

راهنمایی ۹۷: شرایط مسابقه. (بعدی راهنمایی ۲۷، جواب ۲۴)

راهنمایی ۹۸: remove یک پرچم است. remove یک پرچم نیست. (بعدی راهنمایی ۲۲۱، جواب ۳۵)

راهنمایی ۹۹: انسان ها پیش شماره بوستون را بصورت 02126 می نویسند. C++ نظر دیگری دارد. (بعدی راهنمایی ۳۰۸، جواب ۱۵)

راهنمایی ۱۰۰: خروجی نمونه:

```
Area of sample is 0
```

(بعدی راهنمایی ۳۲۶، جواب ۹۳)

راهنمایی ۱۰۱: این یک برنامه قدیمی C بود که بوسیله یک برنامه نویس قدیمی تبدیل به C++ شد. (بعدی راهنمایی ۱۲۰، جواب ۹۸)

راهنمایی ۱۰۲: خروجی چیزی شبیه به این است:

```
474210681171013211611132115111109101329811497105 11
010097109971031011003211211411110311497109321141 01
113117105114101109101110116115441011610410132102 11
...
```

(بعدی راهنمایی ۱۶۰، جواب ۹۹)

راهنمایی ۱۰۳: نتایج وابسته به سیستم هستند. (بعدی راهنمایی ۳۱۴، جواب ۹۰)

راهنمایی ۱۰۴: روی یک سیستم MS – DOS، پیش شماره کلولند[1]، یک عدد منفی است. (بعدی راهنمایی ۲۲۳، جواب ۱۰۴)

راهنمایی ۱۰۵: بهینه ساز می تواند روی این برنامه، کلی کار انجام دهد. (جواب ۸۴)

راهنمایی ۱۰۶: کامپایلر g++ این اخطار را می دهد:

```
comment.cpp:19:35: warning: "/*" within comment
```

(جواب ۹۱)

راهنمایی ۱۰۷: نتایج وابسته به زمان کامپایل هستند. (بعدی راهنمایی ۲۹، جواب ۸۹)

راهنمایی ۱۰۸: بافر چیست؟ (بعدی راهنمایی ۲۶۳، جواب ۶۸)

راهنمایی ۱۰۹: setjmp و longjmp چه کار نمی کنند؟ (جواب ۶۶)

[1] Cleveland

راهنمایی ۱۱۰: یک استثنا گرفته نشده است. در نگاه اول ممکن است این امر غیر ممکن به نظر برسد چرا که فقط یک کلاس استثنا وجود دارد، problem، که ما آنرا می گیریم. حتی اگر ما آنرا نمی گرفتیم، عبارت catch(...) باید هر چیز دیگری را می گرفت. (بعدی راهنمایی ۱۷۳، جواب ۵۵)

راهنمایی ۱۱۱: در برنامه نویسی هیچ چیزی واضح نیست. (جواب ۶۸)

راهنمایی ۱۱۲: با یک خطا از کار می افتد. (بعدی راهنمایی ۲۲۷، جواب ۳۸)

راهنمایی ۱۱۳: 1/3 را بصورت دهدهی بنویسید. (بعدی راهنمایی ۳۰۲، جواب ۵۴)

راهنمایی ۱۱۴: هر دفعه، همان عدد است. (بعدی راهنمایی ۶۶، جواب ۹۷)

راهنمایی ۱۱۵: دو تا if، یک else. بنابراین else به کدام if تعلق دارد؟ (جواب ۳۱)

راهنمایی ۱۱۶: نتیجه وابسته به سیستم است. ممکن است خوش شانس باشید و جواب صحیح را دریافت کنید، یا اینکه جوابهای تصادفی دریافت کنید. (جواب ۵۱)

راهنمایی ۱۱۷: بازه یک short int چیست؟ (جواب ۱)

راهنمایی ۱۱۸: تعریف ماکروی اشتباه. (بعدی راهنمایی ۱۹۰، جواب ۱۱۳)

راهنمایی ۱۱۹: بمباران وابسته به کامپایلر است. روی کامپایلرهای ضعیف، شما به مشکل برمی خورید. کامپایلرهای بهتر، پیغامی خطایی چاپ می کنند مبنی بر اینکه یک تابع مجازی محض[1] را فراخوانی کرده اید. (بعدی راهنمایی ۲۳۷، جواب ۱۰۱)

راهنمایی ۱۲۰: هیچ راهی برای تغییر یک ثابت در یک کلاس وجود ندارد. لذا اگر ما این را از طریق دیباگر اجرا کنیم درمی یابیم که اعداد جادویی، به جای مقادیر مورد انتظار، برابر صفر می باشند. (بعدی راهنمایی ۴، جواب ۹۸)

راهنمایی ۱۲۱: مسأله وابسته به پرچم کامپایل است. (بعدی راهنمایی ۱۴، جواب ۱۱۴)

راهنمایی ۱۲۲: خروجی این است:

11 squared is 121

نه مجذور اعداد ۱ تا ۱۰ آنگونه که برنامه نویس انتظار داشت.

راهنمایی ۱۲۳: چیزی که چاپ می شود، عدد صحیح نیست. (بعدی راهنمایی ۱۴۹، جواب ۸۶)

راهنمایی ۱۲۴: متغیر ch یک کاراکتر است. ch+1 چیست؟ (بعدی راهنمایی ۲۸۳، جواب ۴۵)

راهنمایی ۱۲۵: تعداد دفعاتی که تابع سازنده کپی فراخوانی می شود را بشمارید. (بعدی راهنمایی ۲۳۵، جواب ۱۲)

راهنمایی ۱۲۶: "000" یک دستور مجاز در C++ است. کاملا بی استفاده ولی مجاز. (جواب ۴۴)

[1] Pure Virtual Function

راهنمایی ۱۲۷: بله، I/O بافر شده برای چنین برنامه ای کاربرد دارد. ولی نه آنگونه که در این جا بکار رفته است هرچند که از کتابخانه iostream برای I/O بافرشده استفاده می کنیم. (جواب ۶۵)

راهنمایی ۱۲۸: عمل ضرب چندبار انجام می شود؟ (جواب ۳۹)

راهنمایی ۱۲۹: نتایج وابسته به این هستند که از کدام پرچم های کامپایلر در زمان کامپایل استفاده شده است. (بعدی راهنمایی ۳۱۰، جواب ۹)

راهنمایی ۱۳۰: ماشین های اینتل یک معماری معیوب برای اشاره گر قسمت شده[1] دارند. (بعدی راهنمایی ۲۳۱، جواب ۲۱)

راهنمایی ۱۳۱: نتایج وابسته به کامپایلر هستند. (بعدی راهنمایی ۱۴۱، جواب ۸)

راهنمایی ۱۳۲: تعویض بین پردازه ها می تواند در هر زمانی اتفاق بیفتد. (بعدی راهنمایی ۲۷۶، جواب ۹۲)

راهنمایی ۱۳۳: پیش پردازنده، C++ نیست. (بعدی راهنمایی ۳۶۰، جواب ۴۶)

راهنمایی ۱۳۴: یک روش این جا آمده است:

```
if (i = = 2)
i=1;
else
i = 2;
```

ولی روش سریعتری وجود دارد. (بعدی راهنمایی ۱۴۰، جواب ۴۸)

راهنمایی ۱۳۵: جواب، وابسته به سیستم است. (بعدی راهنمایی ۲۶۴، جواب ۷۰)

راهنمایی ۱۳۶: مبنای هشت. (جواب ۱۵)

راهنمایی ۱۳۷: چه زمانی std::cout مقداردهی اولیه می شود؟ (جواب ۱۱۱)

راهنمایی ۱۳۸: اخطار g++:

```
comment.cpp:11: warning: '/*' within comment
```

(جواب ۶۲)

راهنمایی ۱۳۹: من انتظار داشتم برنامه این را چاپ کند:

```
First 1
First 1
First 1
Second 1
Second 2
Second 3
```

ولی این، آن چیزی نیست که چاپ شد. (بعدی راهنمایی ۲۹۷، جواب ۱۰۲)

راهنمایی ۱۴۰: این جا روش دیگری آمده است:

```
i = (i == 2) ? 1 : 2;
```

ولی روش سریعتری وجود دارد. (بعدی راهنمایی ۲۱۶، جواب ۴۸)

[1] Segmented

راهنمایی ۱۴۱: نتایج می توانند بوسیله سوئیچ های زمان کامپایل در برخی کامپایلرها تغییر کنند. (بعدی راهنمایی ۳۰، جواب ۸)

راهنمایی ۱۴۲: توابع سازنده یک کلاس مشتق شده به این ترتیب فراخوانی می شوند: "پایه، مشتق". توابع مخرب به ترتیب "مشتق، پایه" فراخوانی می شوند. (جواب ۱۰۱)

راهنمایی ۱۴۳: عبارت flags |= CD_SIGNAL; قرار است یک بیت را در پرچم ها مقداردهی کند. اغلب اوقات هم همین کار را می کند. (جواب ۵۲)

راهنمایی ١٤٤: خروجی این است:

(بعدی راهنمایی ۹۱، جواب ۲)

راهنمایی ۱۴۵: نوع متغیری که رد شده است[1]، چیست؟ از دیدگاه تابع، نوع پارامتر چیست؟ (بعدی راهنمایی ۳۱۵، جواب ۷۲)

راهنمایی ۱۴۶: کلاس C++ std::string، حافظه اختصاص می دهد. ولی همچنین آنرا از بین می برد و به دقت طراحی شده است تا مشکل حافظه ایجاد نکند. (بعدی راهنمایی ۳۵۹، جواب ۶۶)

راهنمایی ۱۴۷: برخی کامپایلرها، مانند آنچه که برای این برنامه مورد استفاده قرار گرفته است، اجازه بهینه سازی و دیباگ را می دهند. (بعدی راهنمایی ۱۱، جواب ۸۴)

راهنمایی ۱۴۸: g++ این اخطار را می دهد:

```
/tmp/cckuUagE.o: In function 'std::string::_M_data() const':
/home/sdo/local/include/g++-v3/i586-pc-linux-gnu/bits/gthrsingle.
h(.data+0x0): multiple definition of 'value'
/tmp/ccenmAbd.o(.data+0x0):/home/sdo/local/include/g++-v3/i586-pc-linux-gnu/
bits/gthr-single.h: first defined here
collect2: ld returned 1 exit status
```

(جواب ۵۷)

راهنمایی ۱۴۹: چیزی که چاپ شده، اشاره گر است. (بعدی راهنمایی ۳۴۷، جواب ۸۶)

راهنمایی ۱۵۰: چه زمانی رشته first_name مقداردهی اولیه می شود؟ چه زمانی full_name؟ چه چیزی این ترتیب را ایجاد می کند؟ (جواب ۳)

راهنمایی ۱۵۱: '\n' نشان خط جدید است. (جواب ۳۷)

راهنمایی ۱۵۲: اگر با result کاری نکنیم، پس چرا زحمت محاسبه آنرا به خود بدهیم؟ (جواب ۸۹)

راهنمایی ۱۵۳: اعلان struct info *new_info(void) می تواند به شما راهنمایی کند. (بعدی راهنمایی ۱۰۱، جواب ۹۸)

راهنمایی ١٥٤: چه تعداد اشاره گر وجود دارد؟ به چند چیز اشاره می کنند؟ (بعدی راهنمایی ۲۰۹، جواب ٦٤)

راهنمایی ١٥٥: کامپایلر g++ این اخطار را می دهد:

[1] Passed in

```
equal.cpp: In function 'int main()':
equal.cpp:15: warning: suggest parentheses around assignment used as truth value
```

(بعدی راهنمایی ۲۰۸، جواب ۴۷)

راهنمایی ۱۵۶: خروجی این است:

```
Customer must pay -10
```

(بعدی راهنمایی ۱۱۵، جواب ۳۱)

راهنمایی ۱۵۷: نتایج شما می تواند متفاوت باشد. (بعدی راهنمایی ۷۹، جواب ۱۱۱)

راهنمایی ۱۵۸: SAIL و C تقریبا هیچ سینتکس مشترکی ندارند. دو زبان، کاملا متفاوت هستند ولی خطای تک – کاراکتری باعث خراب شدن هر دو برنامه می شود. (بعدی راهنمایی ۲۲۰، جواب ۵۳)

راهنمایی ۱۵۹: معمولا C++ از "رد با مقدار"[1] برای رد کردن پارامترها استفاده می کند. این بدین معناست که مقدار در زیر برنامه کپی می شود. (بعدی راهنمایی ۲۳۳، جواب ۴۰)

راهنمایی ۱۶۰: خروجی شامل دسته ای از اعداد صحیح است. (جواب ۹۹)

راهنمایی ۱۶۱: من ۳ تا می شمارم. (بعدی راهنمایی ۲۹۳، جواب ۷۱)

راهنمایی ۱۶۲: این خیلی شبیه برنامه ۵۸ است. (بعدی راهنمایی ۱۷۸، جواب ۳۶)

راهنمایی ۱۶۳: i++ چه مقداری برمی گرداند؟ ++i چه مقداری برمی گرداند؟ (بعدی راهنمایی ۹۳، جواب ۸۷)

راهنمایی ۱۶۴: نتایج شما می تواند متفاوت باشد. (بعدی راهنمایی ۱۹، جواب ۸۵)

راهنمایی ۱۶۵: همه می دانند که x=(x*4)/4. این ریاضیات ابتدایی است. (بعدی راهنمایی ۸۰، جواب ۱)

راهنمایی ۱۶۶: اگر فکر می کنید که سوئیچ های زمان کامپایل باید عمل بهینه سازی انجام دهند، در اشتباهید. (بعدی راهنمایی ۳۵۸، جواب ۶۳)

راهنمایی ۱۶۷: با یک عدد علامت دار ۳ بیتی، چه اعدادی می توانند نمایش داده شوند؟ (بعدی راهنمایی ۱۶۹، جواب ۴۲)

راهنمایی ۱۶۸: خروجی این است:

```
Division 5
```

(بعدی راهنمایی ۲۰۲، جواب ۹۱)

راهنمایی ۱۶۹: با یک عدد دو بیتی علامت دار، چند عدد می توانند نمایش داده شوند؟ (بعدی راهنمایی ۵۲، جواب ۴۲)

راهنمایی ۱۷۰: آنچه که چاپ می شود، این است:

```
pi is 1413754136
```

نتایج وابسته به ماشین هستند. (بعدی راهنمایی ۲۰۳، جواب ۱۰)

[1] Pass by value

راهنمایی ۱۷۱: همه آرایه های بیتی، یک اندازه نیستند. (بعدی راهنمایی ۳۵۳، جواب ۵۶)

راهنمایی ۱۷۲: آنچه که می بینید، آنچیزی است که دریافت می کنید. (بعدی راهنمایی ۴۶، جواب ۶۹)

راهنمایی ۱۷۳: استثنا گرفته نشده از کجا آمده است؟ (بعدی راهنمایی ۶۱، جواب ۵۵)

راهنمایی ۱۷۴: gcc این اخطارها را تولید نمود:

```
strcat.c: In function `full_name':
strcat.c:19: warning: implicit declaration of function `strcpy'
strcat.c:20: warning: implicit declaration of function `strcat'
strcat.c:20: warning: passing arg 2 of `strcat' makes pointer from integer witho
ut a cast
strcat.c: In function `main':
strcat.c:28: warning: implicit declaration of function `printf'
```

(جواب ۴۱)

راهنمایی ۱۷۵: اگر برنامه بطور عادی اجرا شود، این عبارت بی استفاده خواهد بود. (بعدی راهنمایی ۲۳۲، جواب ۸۰)

راهنمایی ۱۷۶: نتایج شما ممکن است متفاوت باشد. (بعدی راهنمایی ۲۴، جواب ۱۸)

راهنمایی ۱۷۷: نتایج وابسته به سیستم و کامپایلر هستند. (بعدی راهنمایی ۴۹، جواب ۲۶)

راهنمایی ۱۷۸: این یک نسخه C++ از خطای برنامه ۵۸ است. (جواب ۳۶)

راهنمایی ۱۷۹: عبارت ABORT به نظر یک عبارت می آید ولی اینطور نیست. (جواب ۱۰۵)

راهنمایی ۱۸۰: خروجی این است:

```
-xxxxxxxxxxxxxxx
```

(بعدی راهنمایی ۳۰۳، جواب ۱۹)

راهنمایی ۱۸۱: وقتی scanf، خواندن را متوقف می کند، فایل را در کجا قرار می دهد؟ (جواب ۲۸)

راهنمایی ۱۸۲: برنامه وقتی sscanf را انجام می دهد، از کار می افتد. (بعدی راهنمایی ۲۵۴، جواب ۹۵)

راهنمایی ۱۸۳: آیا I/O بافر شده در این حالت بدرد می خورد؟ (بعدی راهنمایی ۲۱۳، جواب ۶۵)

راهنمایی ۱۸۴: مشکل به استفاده زیاد از delete برمی گردد. (بعدی راهنمایی ۱۸۸، جواب ۱۱۵)

راهنمایی ۱۸۵: ما همیشه بعد از باز کردن فایل، آنرا نمی بندیم. نتیجه این است که واصف[1] ها تمام می شوند. به چند عبارت close(fd) نیاز داریم. (جواب ۶۰)

راهنمایی ۱۸۶: برنامه برای توابعی که به آنها اشاره می شود، از inline استفاده می کند. آیا قسمتی از مشکل می تواند همین باشد؟ (بعدی راهنمایی ۲۱۹، جواب ۷۷)

راهنمایی ۱۸۷: به نظر می رسد که یک نقطه ویرگول بی استفاده در انتهای عبارت زیر وجود دارد:

[1] Descriptor

result=result/*divisor; /* Do divide */;

ولی بدون استفاده نیست. (بعدی راهنمایی ۲۴۵، جواب ۹۱)

راهنمایی ۱۸۸: عملگر delete در تابع سازنده کپی استفاده شده است. چه چیزی حذف می شود؟ (جواب ۱۱۵)

راهنمایی ۱۸۹: چند نمونه از متغیر an_array وجود دارد؟ (بعدی راهنمایی ۳۲۹، جواب ۵۹)

راهنمایی ۱۹۰: آنرا از طریق پیش پردازنده اجرا کنید. (بعدی راهنمایی ۵، جواب ۱۱۳)

راهنمایی ۱۹۱: چه چیزی برگردانده می شود؟ (بعدی راهنمایی ۲۳، جواب ۳۰)

راهنمایی ۱۹۲: خروجی وابسته به سیستم است. (بعدی راهنمایی ۹۰، جواب ۸۸)

راهنمایی ۱۹۳: توگذاری صحیح نیست. (بعدی راهنمایی ۱۲۲، جواب ۳۴)

راهنمایی ۱۹۴: نتایج وابسته به سیستم هستند. (بعدی راهنمایی ۳۲۴، جواب ۱۱۲)

راهنمایی ۱۹۵: prev_ch به تعداد زیادی ایجاد شده است. (جواب ۱۰۶)

راهنمایی ۱۹۶: چه چیزی توسط volatile تغییر می کند؟ (جواب ۶۱)

راهنمایی ۱۹۷: setbuf باعث می شود که داده در کجا قرار بگیرد؟ (بعدی راهنمایی ۸۸، جواب ۸۳)

راهنمایی ۱۹۸: M_PI در math.h بدین صورت تعریف شده است:

```
#define M_PI 3.14159265358979323846 /* pi */
```

(بعدی راهنمایی ۱۶، جواب ۱۰)

راهنمایی ۱۹۹: تابع زیر چه چیزی برمی گرداند؟

```
trouble operator = (const trouble &i_trouble)
```

(بعدی راهنمایی ۳۳۳، جواب ۱۰۹)

راهنمایی ۲۰۰: آن چیزی که مورد انتظار است، چاپ نمی شود. (بعدی راهنمایی ۱۹۲، جواب ۸۸)

راهنمایی ۲۰۱: خروجی:

```
The area is 367
```

(بعدی راهنمایی ۲۵۹، جواب ۲۹)

راهنمایی ۲۰۲: چرا عمل تقسیم انجام نمی شود؟ (بعدی راهنمایی ۱۸۷، جواب ۹۱)

راهنمایی ۲۰۳: کامپایلر g++ این اخطار را می دهد:

```
pi.c: In function 'main':
pi.c:12: warning: int format, double arg (arg 2)
```

(جواب ۱۰)

راهنمایی ۲۰۴: خروجی این است:

```
Y=8
```

(بعدی راهنمایی ۵۴، جواب ۴۹)

راهنمایی ۲۰۵: MAX برابر ۱۰ نیست. (جواب ۱۱۲)

راهنمایی ۲۰۶: تا زمانیکه از نقطه نظر C++ به قضیه بنگریم، خروجی صحیح است. (بعدی راهنمایی ۹۹، جواب ۱۵)

راهنمایی ۲۰۷: کامپایلر g++ این اخطار را می دهد:

```
bit.cpp: In function 'int main()':
bit.cpp:33: warning: comparison is always 0 due to width of bitfield
```

(جواب ۴۲)

راهنمایی ۲۰۸: اجراهای نمونه:

```
$ equal
Enter current balance: 10
You owe 0
$ equal
Enter current balance: 0
You owe 0
$ equal
Enter current balance: -10
You owe 0
```

(بعدی راهنمایی ۲۶۷، جواب ۴۷)

راهنمایی ۲۰۹: فقط یک متغیر و دو اشاره گر وجود دارد. (جواب ۶۴)

راهنمایی ۲۱۰: اعضای ثابت به چه ترتیبی مقداردهی اولیه می شوند؟ (بعدی راهنمایی ۱۰۰، جواب ۹۳)

راهنمایی ۲۱۱: نتایج وابسته به سیستم هستند. (بعدی راهنمایی ۱۲۳، جواب ۸۶)

راهنمایی ۲۱۲: نتایج وابسته به سیستم هستند. (بعدی راهنمایی ۲۲۵، جواب ۱۱۰)

راهنمایی ۲۱۳: آیا در این حالت از I/O بافر شده استفاده شده است؟ (بعدی راهنمایی ۱۲۷، جواب ۶۵)

راهنمایی ۲۱۴: بوضوح، Hello یک بار چاپ شده و دو خط جدید چاپ می شود ولی به همان اندازه واضح است که اگر قرار بود این برنامه کار عاقلانه ای بکند، در این کتاب وجود نداشت. (بعدی راهنمایی ۶۴، جواب ۵۰)

راهنمایی ۲۱۵: به نظر می رسد که یک توضیح در خط ۱۰ و یکی در خط ۱۱ وجود دارد. ولی دقیقا درست نیست. یک توضیح در خطوط ۱۰ و ۱۱ وجود دارد. (بعدی راهنمایی ۱۳۸، جواب ۶۲)

راهنمایی ۲۱۶: سریعترین روش از یک تفریق استفاده کرده و هیچ مقایسه ای انجام نمی دهد. (جواب ۴۸)

راهنمایی ۲۱۷: خروجی برنامه این است:

```
Error: Could not open
oot
ewable
```

(بعدی راهنمایی ۲۴۳، جواب ۳۷)

راهنمایی ۲۱۸: "و" بیتی، "و" منطقی نیست. (جواب ۱۷)

راهنمایی ۲۱۹: برنامه برای توابعی که به آنها اشاره می شود از inline استفاده می کند. آیا قسمتی از مشکل می تواند همین باشد؟ نه. چیزی در این مورد وجود ندارد. C++ این تابع را بخوبی تحت کنترل دارد. (بعدی راهنمایی ۲۷۱، جواب ۷۷)

راهنمایی ۲۲۰: اگر این یک رمز بود، تحلیل فرکانسی اعداد می توانست سرنخی بدهد. در واقع این یک رمز نیست ولی یک تحلیل فرکانسی ارقام می تواند مفید واقع شود. (بعدی راهنمایی ۳۴۱، جواب ۵۳)

راهنمایی ۲۲۱: remove دو چیز است. (بعدی راهنمایی ۱۳، جواب ۳۵)

راهنمایی ۲۲۲: شرایط مسابقه. (بعدی راهنمایی ۱۳۲، جواب ۹۲)

راهنمایی ۲۲۳: اغلب کامپایلرهای یونیکس از ۳۲ بیت برای اعداد صحیح استفاده می کنند. در MS-DOS (نه ویندوز مایکروسافت)، اعداد صحیح معمولا ۱۶ بیتی هستند. (بعدی راهنمایی ۲۵۸، جواب ۱۰۴)

راهنمایی ۲۲۴: اجرای نمونه:

```
% calc
Enter operator and value:+ 5
Total: 5
Enter operator and value:+ 10
Bad operator entered
Total: 5
Enter operator and value:Bad operator entered
Total: 5
Enter operator and value :q
Bad operator entered
Total: 5
Enter operator and value:q
```

(بعدی راهنمایی ۲۵۷، جواب ۲۸)

راهنمایی ۲۲۵: به نظر شما، log_file چگونه مقداردهی اولیه می شود؟ (جواب ۱۱۰)

راهنمایی ۲۲۶: کمتر از سه خطا در این برنامه وجود ندارد که همه طبیعت یکسانی دارند. (بعدی راهنمایی ۷۷، جواب ۶۰)

راهنمایی ۲۲۷: اگر balance برابر صفر باشد چه اتفاقی می افتد؟ (جواب ۳۸)

راهنمایی ۲۲۸: این برنامه روی همه کامپایلرهای شناخته شده C++ کامپایل و اجرا می شود. با این حال غلط است! این امر چگونه ممکن است؟ (بعدی راهنمایی ۳۲۱، جواب ۶۶)

راهنمایی ۲۲۹: بایت بعد از 09 غلط است. (بعدی راهنمایی ۵۸، جواب ۵)

راهنمایی ۲۳۰: تنظیم و لایی گذاری[1]. (بعدی راهنمایی ۲۴۹، جواب ۱۰۳)

راهنمایی ۲۳۱: روی ماشین های اینتل، در برخی مدل های حافظه، کامپایلر کدی تولید می کند که فقط قسمت آدرس یک اشاره گر را دستکاری می کند و با سگمنت کاری ندارد.

[1] Alignment and padding

راهنمایی ۲۳۲: این عبارت فقط وقتی مفید است که برنامه را در یک دیباگر تعاملی[1] اجرا کنید. (بعدی راهنمایی ۳۰۲، جواب ۸۱)

راهنمایی ۲۳۳: کپی کردن یک متغیر ostream چه معنی دارد؟ (جواب ۴۰)

راهنمایی ۲۳۴: 0a کد اسکی line feed است. (بعدی راهنمایی ۲، جواب ۵)

راهنمایی ۲۳۵: تابع سازنده کپی، دو جا فراخوانی شده است. (جواب ۱۲)

راهنمایی ۲۳۶: var_array::~var_array چندبار فراخوانی می شود؟ (بعدی راهنمایی ۲۸۶، جواب ۵۹)

راهنمایی ۲۳۷: کامپایلر مانع از این می شود که یک تابع مجازی محض را فراخوانی کنید. شما نمی توانید یک نمونه از یک کلاس مجرد تعریف کنید و هر کلاس پایه ای باید یک نسخه مشتق شده داشته باشد که همه توابع مجازی محض را تعریف کند. این بدین معناست که هر تابع مجازی محض، یک تعریف واقعی در کلاس پایه خواهد داشت.

بنابراین چگونه یکی را فراخوانی کردیم اگر می دانیم که یک پیاده سازی برای تابع مجازی باید در کلاس مشتق وجود داشته باشد؟ (بعدی راهنمایی ۱۴۲، جواب ۱۰۱)

راهنمایی ۲۳۸: عقل سلیم می گوید که اگر آرایه ای را بصورت int array[5] تعریف می کنیم، عناصر آن array[1], array[2], array[3], array[4], array[5] می باشند. با این حال عقل سلیم چیزی از برنامه نویسی نمی داند. (جواب ۹۰)

راهنمایی ۲۳۹: در زیر، نسخه شانزده شانزدهی خروجی MS-DOS آمده است:

```
000000 00 01 02 03 04 05 06 07 08 09 0d 0a 0b 0c 0d 0e
000010 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e
000020 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e
000030 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e
000040 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e
000050 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e
000060 5f 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e
000070 6f 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e
000080 7f
```

(بعدی راهنمایی ۲۲۹، جواب ۵)

راهنمایی ۲۴۰: کاما، یک عملگر C++ است. (بعدی راهنمایی ۴۷، جواب ۴۴)

راهنمایی ۲۴۱: تعویض بین ریسمان ها[2] بین هر دو خطی مانند دو خط زیر می تواند رخ دهد:

```
++count; // We've got a new character
*in_ptr = ch;// Store the character
```

(جواب ۹۲)

راهنمایی ۲۴۲: کامپایلر g++، این اخطار را می دهد:

```
def.cpp: In function 'int main()':
def.cpp:19: warning: label 'defualt' defined but not used
```

(جواب ۶۷)

[1] Interactive
[2] Thread Switching

راهنمایی ۲۴۳: در یونیکس، name دارای طول ۱۵ کاراکتر است. در MS-DOS، طول آن ۱۲ کاراکتر است. (بعدی راهنمایی ۱۵۱، جواب ۳۷)

راهنمایی ۲۴۴: با استفاده از g++ وقتی که برنامه این طور کامپایل شود، کار می کند:

```
g++ -g -Wall -o last last.cpp first.cpp
```

ولی وقتی بصورت زیر کامپایل شود، کار نمی کند:

```
g++ -g -Wall -o last first.cpp last.cpp
```

(بعدی راهنمایی ۱۵۰، جواب ۳)

راهنمایی ۲۴۵: اگر ادیتور شما دارای پررنگ کردن سینتکس[1] است، ببینید وقتی که این کد را درون آن قرار می دهید، چه اتفاقی می افتد. (بعدی راهنمایی ۳۳۸، جواب ۹۱)

راهنمایی ۲۴۶: واضح است که جواب، ۳ است. (++i ۲ است و یک ++ دیگر آنرا تبدیل به ۳ می کند). ولی در برنامه نویسی، چیزی واضح نیست. (بعدی راهنمایی ۳۷، جواب ۸۷)

راهنمایی ۲۴۷: فاصله گذاری. (بعدی راهنمایی ۳۲۵، جواب ۲۳)

راهنمایی ۲۴۸: دو اعلان متغیر در این برنامه وجود دارد. (بعدی راهنمایی ۸۳، جواب ۵۷)

راهنمایی ۲۴۹: ۶ بر ۴ بخش پذیر است. (جواب ۱۰۳)

راهنمایی ۲۵۰: تابع دوباره تعریف شده new باید کار کند چون واضح است که تمام آرایه های بیتی دارای اندازه یکسان هستند. (بعدی راهنمایی ۱۷۱، جواب ۵۶)

راهنمایی ۲۵۱: پایان خود را ببینید. (جواب ۴۳)

راهنمایی ۲۵۲: فراخوانی سیستم fork یک پردازه جدا با حافظه جدا ایجاد می کند که شامل داده بافر شده printf نیز می شود. (جواب ۵۰)

راهنمایی ۲۵۳: افراد عادی، اینگونه تا ۵ می شمارند: "۱و ۲و ۳و ۴و ۵". برنامه نویس های C++ می گویند: "۰و ۱و ۲و ۳و ۴". (بعدی راهنمایی ۲۳۸، جواب ۹۰)

راهنمایی ۲۵۴: اخطار gcc:

```
calc2.c: In function 'main':
calc2.c:24: warning: format argument is not a pointer (arg 3)
calc2.c:24: warning: format argument is not a pointer (arg 4)
```

(جواب ۹۵)

راهنمایی ۲۵۵: پیش پردازنده، سینتکس C++ را نمی فهمد. (بعدی راهنمایی ۲۹۵، جواب ۷۸)

راهنمایی ۲۵۶: اگر بخواهید برنامه را دیباگ کنید، مشکل حل می شود. (بعدی راهنمایی ۱۲۱، جواب ۱۱۴)

راهنمایی ۲۵۷: نتایج شما ممکن است متفاوت باشد. (بعدی راهنمایی ۱۸۱، جواب ۲۸)

[1] Syntax Highlighting

راهنمایی ۲۵۸: یک عدد صحیح ۱۶ بیتی، می تواند از ۳۲۷۶۸- تا ۳۲۷۶۷ تغییر کند. (جواب ۱۰۴)

راهنمایی ۲۵۹: جواب برابر ۳۶۷ (۳۳۰ + ۳۷) است. (جواب ۲۹)

راهنمایی ۲۶۰: strcmp مقدار true/false برنمی گرداند. (جواب ۷۶)

راهنمایی ۲۶۱: ;'\0'=char prev_ch وقتی که prev_ch ایجاد می شود، اجرا می گردد. (بعدی راهنمایی ۱۹۵، جواب ۱۰۶)

راهنمایی ۲۶۲: عبارت (0 !=n2) if مقدار n2 را تغییر می دهد. (جواب ۲۵)

راهنمایی ۲۶۳: طراحان یونیکس با دانش بی نهایت خود، برای تقسیم عدد صحیح بر صفر، این پیغام را صادر می کنند:

Floating exception (core dumped)

(بعدی راهنمایی ۹۲، جواب ۶۸)

راهنمایی ۲۶۴: بعضی سیستم ها، این اجازه را می دهند که اشاره به NULL را از بین ببرید؛ دیگر سیستم ها این اجازه را نمی دهند. (جواب ۷۰)

راهنمایی ۲۶۵: در این کتاب نیست! (بعدی راهنمایی ۷۲، جواب ۳۶)

راهنمایی ۲۶۶: عبارت 2 >> x واقعا برابر ۴ است. با این حال، ما از این عبارت در این برنامه استفاده نمی کنیم. (بعدی راهنمایی ۲۰۴، جواب ۴۹)

راهنمایی ۲۶۷: عبارت (0 = amount) if مقدار amount و صفر را مقایسه نمی کند. (جواب ۴۷)

راهنمایی ۲۶۸: خروجی این برنامه این است:

```
The number of sheep is: 100
The number of sheep is: 1000
The number of sheep is: -6384
```

(بعدی راهنمایی ۱۱۷، جواب ۱)

راهنمایی ۲۶۹: []char =! char * (بعدی راهنمایی ۲۵، جواب ۷)

راهنمایی ۲۷۰: خروجی این است:

Size is 25

نه Size is 20 آنگونه که برنامه نویس انتظار داشت. (جواب ۴)

راهنمایی ۲۷۱: جواب وابسته به سیستم است. (بعدی راهنمایی ۳۲، جواب ۷۷)

راهنمایی ۲۷۲: اگر کشتن برنامه نویس ممکن نباشد، به طریقی باید آگاه شود که روش خوب برنامه نوشتن چگونه است. (بعدی راهنمایی ۱۶۳، جواب ۸۷)

راهنمایی ۲۷۳: خروجی، روی یک سیستم اینگونه است:

```
3 squared is 9
5 squared is 25
7 squared is 49
9 squared is 81
```

```
11 squared is 121
```

(جواب ۸۸)

راهنمایی ۲۷۴: عبارت switch یک حالت default ندارد فقط این طور به نظر می رسد. (بعدی راهنمایی ۲۴۲، جواب ۶۷)

راهنمایی ۲۷۵: خط ۱۶ هیچ مشکلی ندارد. ما برای گمراه ساختن شما این را گفتیم. (بعدی راهنمایی ۴۰، جواب ۷۹)

راهنمایی ۲۷۶: reader این دو خط را انجام می دهد:

```
++count; // We've got a new character
*in_ptr = ch;// Store the character
```

(بعدی راهنمایی ۲۴۱، جواب ۹۲)

راهنمایی ۲۷۷: اخطار gcc:

```
two.c: In function 'main':
two.c:11: warning: too few arguments for format
two.c:9: warning: unused variable `answer'
```

(جواب ۸۵)

راهنمایی ۲۷۸: بعضی کامپایلرها، سوئیچی دارند که رفتار برنامه را تغییر می دهد. سوئیچ چیزی را درست نمی کند ولی رفتار برنامه را تغییر می دهد. (بعدی راهنمایی ۱۶۶، جواب ۶۳)

راهنمایی ۲۷۹: بیشتر از یک i1 وجود دارد. (بعدی راهنمایی ۹۶، جواب ۹۴)

راهنمایی ۲۸۰: تفاوت بین "و" و "و و". (بعدی راهنمایی ۷۶، جواب ۱۷)

راهنمایی ۲۸۱: برنامه این را چاپ می کند:

```
Result is 0
```

(جواب ۲۷)

راهنمایی ۲۸۲: اگر کد را بررسی کنید، من همیشه مطئنم که متغیر data را قبل از اینکه مقدار آنرا تغییر دهم، delete می کنم. (بعدی راهنمایی ۱۸۴، جواب ۱۱۵)

راهنمایی ۲۸۳: خروجی این است:

```
A6667
```

(بعدی راهنمایی ۶۲، جواب ۴۵)

راهنمایی ۲۸۴: عبارت ;1.0- یک عبارت خوب C++ است. با اینکه کاملا بی مصرف است، ولی کاملا مجاز است. (بعدی راهنمایی ۱۲، جواب ۸۲)

راهنمایی ۲۸۵: از آنجا که هیچ سرآمدی[1] وجود ندارد، هیچ پیش الگویی برای توابع استاندارد نداریم؛ آنها بصورت ضمنی اعلان شده اند. (بعدی راهنمایی ۸۱، جواب ۴۱)

راهنمایی ۲۸۶: var_array::~var_array دو بار فراخوانی شده است. (جواب ۵۹)

[1] Header

راهنمایی ۲۸۷: اگر ادیتوری با رنگ آمیزی سینتکس دارید، base را با یک رنگ و height را با رنگ دیگری نمایش خواهد داد. (بعدی راهنمایی ۲۱۵، جواب ۶۲)

راهنمایی ۲۸۸: حتی اگر فایل موجود باشد و اجازه خواندن آنرا هم داشته باشیم، open با شکست مواجه می شود. (بعدی راهنمایی ۳۰۶، جواب ۶۰)

راهنمایی ۲۸۹: عبارت int &i=3+4; مجاز است ولی نگران نباشید، ما از آن استفاده نمی کنیم – حداقل نه به این شکل. (جواب ۲۲)

راهنمایی ۲۹۰: خروجی این است:

```
Width is too small
area(10, 10) = 100
```

چیزی که برنامه نویس انتظار دارد این است:

```
Width is too small
area(10, 50) = 500
```

(بعدی راهنمایی ۶۷، جواب ۱۳)

راهنمایی ۲۹۱: تابع سازنده کپی بیش از آنچه که فکر می کنید، فراخوانی می شود. (بعدی راهنمایی ۳۱۶، جواب ۱۰۹)

راهنمایی ۲۹۲: به یاد آورید که "1" برابر "1.0" نیست. (بعدی راهنمایی ۲۸۱، جواب ۲۷)

راهنمایی ۲۹۳: تعداد بایت ها در داده struct ایجاد کننده دو مشکل است. (بعدی راهنمایی ۵۱، جواب ۷۱)

راهنمایی ۲۹۴: strcmp می تواند یک تازه کار را به اشتباه بیندازد. (بعدی راهنمایی ۲۶۰، جواب ۷۶)

راهنمایی ۲۹۵: خروجی را از طریق پیش پردازنده اجرا کنید. (جواب ۷۸)

راهنمایی ۲۹۶: خروجی نمونه:

```
Stack 0 has 1 elements
Stack 1 has 100 elements
Stack 2 has 134516168 elements
Stack 3 has 134525376 elements
Stack 4 has 4 elements
```

(بعدی راهنمایی ۱۴۵، جواب ۷۲)

راهنمایی ۲۹۷: مقدار ++i چیست؟ مقدار i++ چیست؟ (جواب ۱۰۲)

راهنمایی ۲۹۸: آنها متفاوتند. (بعدی راهنمایی ۲۵۵، جواب ۷۸)

راهنمایی ۲۹۹: به نظر می رسد اعداد در مبنای هشت باشند. (جواب ۵۳)

راهنمایی ۳۰۰: تغییر دادن یک ثابت درون یک کلاس، غیر ممکن است ولی این برنامه این کار را می کند. ایجاد یک کلاس بدون فراخوانی تابع سازنده غیر ممکن است ولی این برنامه این کار را می کند. (جواب ۹۸)

راهنمایی ۳۰۱: این برنامه دارای خروجی زیر است:

```
parity
-break
```

xon
-rts

(جواب ۱۰۸)

راهنمایی ۳۰۲: 1/3 را سه بار بصورت دهدهی در یک ستون بنویسید. حالا آنها را با هم جمع کنید. (جواب ۵۴)

راهنمایی ۳۰۳: ((b)0000 0000 1000) 0x8000 برابر (15>>1) است. این مقدار صحیح و آنچیزی است که برنامه نویس انتظار داشت. (بعدی راهنمایی ۸۴، جواب ۱۹)

راهنمایی ۳۰۴: توگذاری غلط است. (بعدی راهنمایی ۲۷۰، جواب ۴)

راهنمایی ۳۰۵: ایجاد تابع new توسط خودتان می تواند سرعت کار را بالا ببرد – اگر که این کار را درست انجام دهید. (بعدی راهنمایی ۲۵۰، جواب ۵۶)

راهنمایی ۳۰۶: open با یک خطای EMFILE با شکست مواجه می شود. (پردازه دارای بیشترین تعداد فایل های باز است). (بعدی راهنمایی ۱۸۵، جواب ۶۰)

راهنمایی ۳۰۷: رشته "Hello World!/n" دارای ۱۴ کاراکتر است. (جواب ۶۹)

راهنمایی ۳۰۸: برنامه این را چاپ می کند:

San Diego 92126
Boston 01110

(بعدی راهنمایی ۱۳۶، جواب ۱۵)

راهنمایی ۳۰۹: برنامه نویس وقتی دارد داده 500# را می خواند فکر می کند اتفاق احمقانه ای دارد می افتد. (جواب ۸۱)

راهنمایی ۳۱۰: in_port_ptr* چندبار خوانده می شود؟ (بعدی راهنمایی ۳۵۶، جواب ۹)

راهنمایی ۳۱۱: ساختمان داده ای که استفاده شد، یک درخت دودویی نامتوازن است. (بعدی راهنمایی ۳۲۳، جواب ۷۴)

راهنمایی ۳۱۲: من نمی دانستم که شما می توانید مقایسه های سه گانه مانند a<b>c انجام دهید. (بعدی راهنمایی ۱۸، جواب ۸۰)

راهنمایی ۳۱۳: یک کاراکتر، ۸ بیت دارد. شماره های آنها چیست؟ (جواب ۱۱)

راهنمایی ۳۱۴: برخی مواقع وقتی برنامه را اجرا می کنید، جواب غلط دریافت می کنید، برخی مواقع هم با تخلف سگمنت بندی[1] همه چیز را نابود می کنید (کاربران ویندوز یک UAE(Unexpected Application Error) دریافت می کنند و برخی مواقع هم همه چیز درست کار می کند. (بعدی راهنمایی ۲۵۳، جواب ۹۰)

راهنمایی ۳۱۵: اندازه عنصر آرایه ها چقدر است؟ (جواب ۷۲) sizeof(stack) != sizeof(safe_stack)

راهنمایی ۳۱۶: خروجی نمونه:

Copy Constructor called
= operator called

[1] Segmentation Violation

```
Copy Constructor called
= operator called
Copy Constructor called
= operator called
...
```

(بعدی راهنمایی ۱۹۹، جواب ۱۰۹)

راهنمایی ۳۱۷: این برنامه سعی دارد تا درست بعد از پاک کردن داده، آنرا کپی کند. (بعدی راهنمایی ۴۳، جواب ۷۵)

راهنمایی ۳۱۸: آرگومان های memset کدامند؟ (بعدی راهنمایی ۳۳۷، جواب ۲۰)

راهنمایی ۳۱۹: نتایج وابسته به سیستم هستند. (بعدی راهنمایی ۱۹۱، جواب ۳۰)

راهنمایی ۳۲۰: دلیلی وجود دارد که هیچ سرآمدی را در این برنامه نگذاشتیم. (بعدی راهنمایی ۲۸۵، جواب ۴۱)

راهنمایی ۳۲۱: این برنامه غیر استاندارد است. (بعدی راهنمایی ۶۸، جواب ۶)

راهنمایی ۳۲۲: این همان برنامه ۵۸ نیست؟ نه، در واقع این برنامه ای است که یک بهبود در آن داده اید. به اعلان زیبای ایستا در خط ۲۲ توجه کنید. (ولی هنوز هم مشکل دارد) (بعدی راهنمایی ۱۵۴، جواب ۶۴)

راهنمایی ۳۲۳: ساختمان داده مورد استفاده، یک درخت دودویی بسیار نامتوازن است. (جواب ۷۴)

راهنمایی ۳۲۴: خروجی پیش پردازنده را بررسی کنید. (بعدی راهنمایی ۳۸، جواب ۱۱۲)

راهنمایی ۳۲۵: این برنامه به سه چیز احتیاج دارد: "فاصله"، "فاصله" و "فاصله". (جواب ۲۳)

راهنمایی ۳۲۶: خروجی g++ این اخطار را می دهد:

```
rect.cpp: In constructor `rectangle::rectangle(int, int)':
rect.cpp:20: warning: member initializers for 'const int rectangle::height'
rect.cpp:18: warning: and 'const int rectangle::area'
rect.cpp:31: warning: will be re-ordered to match declaration order
```

(جواب ۹۳)

راهنمایی ۳۲۷: پیش پردازنده از قوانین خود پیروی می کند. (بعدی راهنمایی ۲۰۱، جواب ۲۹)

راهنمایی ۳۲۸: out_file چیست؟ (بعدی راهنمایی ۷۵، جواب ۴۰)

راهنمایی ۳۲۹: وقتی store_it درحال اجرا است، دو نمونه از کلاس var_array وجود دارد. (بعدی راهنمایی ۳۵۵، جواب ۵۹)

راهنمایی ۳۳۰: وقتی تابعی در کلاس مشتق تعریف نشده است، C++ از کلاس پایه برای آن استفاده می کند. خب پس چه چیزی مانع این می شود که C++، base::print_it(int) را فراخوانی نکند؟ (جواب ۵۸)

راهنمایی ۳۳۱: ما نتیجه ضرب را ۱۸۶۳ بار در result ذخیره می کنیم. بنابراین حلقه درحال اجراست. (بعدی راهنمایی ۱۲۸، جواب ۳۹)

راهنمایی ۳۳۲: شماره سمت چپ ترین بیت یک کلمه ۱۶ – بیتی چیست؟ (بعدی راهنمایی ۱۴۴، جواب ۲)

راهنمایی ۳۳۳: تابع operator = چگونه نتیجه خود را برمی گرداند؟ (جواب ۱۰۹)

راهنمایی ۳۳۴: عملگر انتصاب، مشکل دارد. (بعدی راهنمایی ۳۵۷، جواب ۱۴)

راهنمایی ۳۳۵: شما نمی توانید. (بعدی راهنمایی ۲۴۰، جواب ۴۴)

راهنمایی ۳۳۶: چیزهایی و اشاره گرهایی به این چیزها وجود دارد. (بعدی راهنمایی ۱۹۶، جواب ۶۱)

راهنمایی ۳۳۷: sizeof(array) یک کاراکتر نیست و '\0' هم عدد صحیح نیست. C++ آنقدر باهوش نیست که متوجه این قضیه شود. (جواب ۲۰)

راهنمایی ۳۳۸: توضیحات با */ شروع شده و به /* ختم می شوند. (بعدی راهنمایی ۱۰۶، جواب ۹۱)

راهنمایی ۳۳۹: این دومین باری است که یک استثنا می اندازیم و مشکل همین جاست. (بعدی راهنمایی ۳۴۵، جواب ۵۵)

راهنمایی ۳۴۰: بیش از آنچه که فکر می کنید مسأله به اسم برمی گردد. (بعدی راهنمایی ۷۱، جواب ۳۳)

راهنمایی ۳۴۱: اگر تحلیل فرکانسی را انجام دادید، دریافته اید که ارقام ۸ و ۹ در خروجی وجود ندارند. (بعدی راهنمایی ۲۹۹، جواب ۵۳)

راهنمایی ۳۴۲: 12*34=408. همیشه برابر ۴۰۸ است. همه این را می دانند از جمله کامپایلر. (جواب ۱۶)

راهنمایی ۳۴۳: تنظیم. (بعدی راهنمایی ۲۳۰، جواب ۱۰۳)

راهنمایی ۳۴۴: چرا کامپیوتر، تست 1>c را انجام می دهد؟ (جواب ۸۰)

راهنمایی ۳۴۵: دومین استثنا از تابع مخرب پشته انداخته شده است. (جواب ۵۵)

راهنمایی ۳۴۶: مشکل در خط ۵ وجود دارد. (جواب ۷۹)

راهنمایی ۳۴۷: کامپایلر g++ این اخطار را می دهد:

```
array2.cpp: In function 'int main()':
array2.cpp:17: warning: left-hand operand of comma expression has no effect
```

(بعدی راهنمایی ۱۷، جواب ۸۶)

راهنمایی ۳۴۸: matrix[2] یک اشاره گر است. (جواب ۸۶)

راهنمایی ۳۴۹: نسخه MS-DOS یک کاراکتر درج می کند. (بعدی راهنمایی ۲۳۹، جواب ۵)

راهنمایی ۳۵۰: مسأله وابسته به کامپایلر است. (بعدی راهنمایی ۲۵۶، جواب ۱۱۴)

راهنمایی ۳۵۱: برنامه این را چاپ می کند:

```
At least one number is zero.
```

(بعدی راهنمایی ۲۸۰، جواب ۱۷)

راهنمایی ۳۵۲: برنامه دقت ۶۴ بیت را گزارش می دهد. (بعدی راهنمایی ۵۷، جواب ۷۳)

راهنمایی ۳۵۳: دلیلی وجود دارد که تابع بازتعریف شده new، size را بعنوان یک پارامتر رد می کند. (جواب ۵۶)

راهنمایی ۳۵۴: برنامه چاپ می کند:

```
2 is prime
3 is prime
5 is prime
7 is prime
```

ما انتظار یک سری پیغام را داشتیم که بگوید ۴و ۶و ۸و ۹، اعداد اول نیستند. ولی بنا به دلایلی آن پیغام ها ناپدید شدند. (بعدی راهنمایی ۲۷۴، جواب ۶۷)

راهنمایی ۳۵۵: تابع سازنده کپی چگونه پیاده سازی شده است؟ (بعدی راهنمایی ۲۳۶، جواب ۵۹)

راهنمایی ۳۵۶: چندبار باید in_port_ptr* خوانده شود تا برنامه کار کند؟ (حداقل در ظاهر) (جواب ۹)

راهنمایی ۳۵۷: توسط عبارت save_queue = a_queue چه چیزی مقداردهی می شود؟ (جواب ۱۴)

راهنمایی ۳۵۸: سوئیچ های زمان کامپایل وظیفه تبدیل بین char و int را بر عهده دارند. (جواب ۶۳)

راهنمایی ۳۵۹: setjmp و longjmp چه کار می کنند؟ (بعدی راهنمایی ۱۰۹، جواب ۶۶)

راهنمایی ۳۶۰: آنرا از طریق پیش پردازنده اجرا کنید. (جواب ۴۶)

راهنمایی ۳۶۱: هی استیو! نمی توانی این مشکل را حل کنی؟ (بعدی راهنمایی ۲۶۵، جواب ۳۶)

قسمت سوم: جوابها

جواب ۱: مسأله این جاست که یک گله بزرگ، ۱۰۰۰۰ گوسفند دارد. یعنی ۴۰۰۰۰ پا. بزرگترین عددی که می توانید در یک short int قرار دهید، ۳۲۷۶۷ است. این عدد کوچکتر از ۴۰۰۰۰ است بنابراین (۱۰۰۰۰ * ۴) باعث یک سرریز می شود که درنتیجه خروجی اشتباه تولید می کند.

جواب ۲: مسأله این است که عبارت :

```
// The bit we are printing now
short int bit = (1<<16);
```

بیت متغیر را با 1 0000 0000 0000 0000(b) مقداردهی نمی کند. درعوض، مقدار آنرا برابر 0000 0000 0000 0000(b) قرار می دهد. متأسفانه، آن نمی تواند ۱۷ بیت را در خود نگهدارد لذا نتیجه این است که مقدارش برابر صفر می شود. از آنجا که مقدارش صفر است، عبارت ارزیابی همیشه مقدار غلط دارد و نتیجه این می شود: ----------

جواب ۳: کلاس های سراسری قبل از main مقداردهی اولیه می شوند. ترتیب این کار مشخص نیست. علی الخصوص، هیچ تضمینی وجود ندارد که first_name قبل از اینکه مورد استفاده قرار بگیرد، مقداردهی شود. بنابراین اگر کامپایلر، ترتیب غلطی را انتخاب کند، برنامه خروجی غلطی را می دهد یا کلا از کار می افتد.

جواب ۴: برنامه نویس فکر می کرد که دو عبارت را درون if قرار داده است ولی آکولادها را فراموش کرده بود. بنابراین عبارت:

```
if (size > MAX)
std::cout << "Size is too large\n";
size = MAX;
```

در واقع به این صورت است:

```
if (size > MAX)
        std::cout << "Size is too large\n";
size = MAX;
```

و آنچه که برنامه نویس باید می نوشت این بود:

```
if (size > MAX)
{
        std::cout << "Size is too large\n";
        size = MAX;
}
```

جواب ۵: مشکل این است که نوع فایل را از نوع دودویی (ios::bin) مشخص نکردیم. کتابخانه زمان اجرای ویندوز مایکروسافت، کاراکتر خروجی را دست می زند و قبل از هر <line-feed(0xA)> یک <carriage-return(0xD)> درج می کند. این امر، 0D اضافی که قبل از کاراکتر 0A در فایل قرار دارد را توجیه می کند.

جواب ۶: مشکل خط void main() 6 است. تابع main() یک تابع void نیست. یک int است. این تابع یک کد خروجی را به سیستم عامل برمی گرداند. یک "Hello World" درست می توانست بدین شکل باشد:

```
1 /************************************************
2 * The "standard" hello world program. *
3 ************************************************/
4 #include <ostream>
5
6 int main()
7 {
8 std::cout << "Hello world!\n";
```

```
9 return (0);
10 }
```

وقتی همسر من برنامه نویسی را شروع کرد، این اولین برنامه ای بود که یاد گرفت (نسخه void). من void را به int تغییر دادم و او تمرین خود را تحویل داد. معلم حل تمرین، این را غلط گرفت و به حالت اولیه برگرداند.

نیازی به گفتن نیست که من از این قضیه راضی نبودم و یک نامه خیلی تحقیرآمیز برای او نوشته و متذکر شدم که main از نوع int بوده است و با یادآوری استاندارد C++ حرف خودم را به اثبات رساندم. او به نامه من جواب داد و خیلی هم خوشحال شده بود.

جواب ۷: مشکل این است که sub.cpp، str را بصورت یک آرایه کاراکتری تعریف می کند ([]char). عبارت extern در main.cpp، str را بصورت اشاره گر کاراکتری تعریف می کند (* char). تقریبا همیشه در C++، آرایه های کاراکتری و اشاره گرهای کاراکتری قابل تعویض می باشند. این یکی از معدود حالاتی است که این قضیه صدق نمی کند. در این حالت، برنامه main فکر می کند که str یک اشاره گر کاراکتری است، لذا به آن مکان رفته و اولین چهار بایت را بعنوان آدرس می خواند. اولین چهار بایت برابر "Hell" است که یک آدرس نمی باشد و بنابراین برنامه از کار می افتد.

همیشه extern ها را در یک فایل سرآمد تعریف کنید. این سرآمد باید همیشه در ماژولی که این قلم تعریف می شود و در تمام ماژول هایی که از آن استفاده می کنند، include شود.

جواب ۸: مسأله این است که ch می تواند یک کاراکتر علامت دار باشد. یعنی اگر ch برابر 0xFF باشد، وقتی که به منظور مقایسه به یک عدد صحیح علامت دار تبدیل می شود، خواهیم داشت int(ch) = -1 (0xFFFFFFF). که این مقدار برابر 0xFF نیست و مقایسه درست کار نمی کند.

وقتی از متغیرهای کاراکتری برای نگهداری اعداد استفاده می کنید، مراقب باشید. آنها ممکن است آن کاری را که شما انتظار دارید انجام ندهند.

جواب ۹: مشکل این است که بهینه ساز، به برنامه نگاه می کند و می بیند که ما in_port_ptr* را سه بار می خوانیم و سپس نتیجه را دور می اندازیم. لذا بهینه ساز درمی یابد که می تواند برنامه را بهبود بخشد و با حذف خطوط ۲۰و ۲۱و ۲۲ همان نتایج را تولید کند. راه حل این است که اشاره گرهای پورت را بصورت volatile تعریف کنید. در برنامه ۱۰۷ ما این کار را کردیم ولی یک چیزی درست نیست.

جواب ۱۰: پاسخ این است که فرمت (%d) در printd به نوع پارامتر (double) نمی خورد. برنامه نویس باید اینگونه می نوشت:

```
12 printf("pi is %f\n", M_PI);
```

جواب ۱۱: یک کاراکتر، ۸ بیت دارد که از ۰ تا ۷ شماره گذاری شده اند. این بیت ها را می توان با اعداد (1<<0) تا (1<<7) نشان داد. شماره بیت ۸ نداریم لذا عبارت:

```
privs |= P_BACKUP; // P_BACKUP = (1 << 8)
```

کاری نمی کند، چون بیتی در خارج از محدوده کاراکتر را مقداردهی می کند. نتیجه این است که فقط امتیاز مدیریت، مقداردهی می شود.

جواب ۱۲: فراخوانی تابع operator = یک پارامتر از نوع data_holder می گیرد. این نوع پارامتر، یک پارامتر فراخوانی با مقدار است، لذا تابع سازه کپی فراخوانی می شود. برنامه نویسی که تابع سازنده کپی را می نوشت، تصمیم گرفت که میانبر بزند و از عملگر = برای انجام کپی استفاده کند. لذا operator = تابع سازنده کپی

را فراخوانی می کند که به نوبه خود، operator = را فراخوانی می کند که این هم دوباره تابع سازنده کپی را فراخوانی می کند... و این حلقه آنقدر ادامه پیدا می کند تا پشته پر شود.

تابع operator = باید یک ارجاع ثابت را بعنوان نوع پارامتر خود بگیرد:

```
data_holder &operator = (
        const data_holder &old_data_holder) {
```

همچنین باید یک ارجاع را برگرداند.

در موقع ارسال پارامترها، تا حد امکان از ارجاع های const استفاده کنید. این کار از هزینه اضافی انجام دادن یک کپی توسط یک پارامتر فراخوانی با مقدار جلوگیری می کند.

جواب ۱۳: مشکل در عبارت if است. در اولی:

```
if (width < MIN) {
        std::cout << "Width is too small\n";
        width = MIN;
```

برنامه نویس فراموش کرد که آکولاد بسته را بگذارد. البته مشکلی ایجاد نشد چون این قضیه با فراموش کردن یک آکولاد باز در if بعدی جبران شد:

```
if (height < MIN)
        std::cout << "Height is too small\n";
        height = MIN;
}
```

اگر برنامه را درست توگذاری کنیم، می توانیم پی به مشکل ببریم:

```
if (width < MIN) {
        std::cout << "Width is too small\n";
        width = MIN;

        if (height < MIN)
                std::cout << "Height is too small\n";
        height = MIN;
}
```

آنچه که برنامه نویس می بایست می نوشت این است:

```
if (width < MIN) {
        std::cout << "Width is too small\n";
        width = MIN;
}

if (height < MIN) {
        std::cout << "Height is too small\n";
        height = MIN;
}
```

جواب ۱۴: عبارت save_queue = a_queue صفی به طول ۳۰ را در صفی به طول ۲۰ کپی می کند. بعبارت دیگر، عملگر انتصاب (آنگونه که پیاده سازی شده است) به ما این اجازه را می دهد که صفهایی با اندازه متفاوت را کپی کنیم. ما نباید مجاز به این کار باشیم.

برای حل این مشکل، چهار راه وجود دارد:

۱. از کلاس صف STL استفاده کنیم.
۲. عملگر انتصاب را private کنیم (و اجازه هیچ انتصابی را ندهیم).
۳. عملگر انتصاب را طوری تغییر دهیم که در صورت یکسان نبودن اندازه صف ها، استثنا بیندازد.
۴. کلاس queue را طوری تغییر دهیم که بتوان صفهای با اندازه متفاوت را به هم انتصاب کرد.

جواب ۱۵: ثابت 02126 در مبنای هشت است چون اولین رقم آن صفر است. بنابراین در C++، 02126 (هشت هشتی) برابر 1110 (دهدهی) است و برابر پیش شماره بوستون نمی باشد.

جواب ۱۶: مشکل این جاست که کامپایلر می داند که 34*12 برابر چه می شود، لذا بجای انجام عمل ضرب، عبارت را بهینه سازی می کند و آنرا تبدیل می کند به:

```
18 result = 408;
```

از آنجا که عمل ضرب انجام نمی شود، زمانبندی کار نمی کند. برنامه ۱۰۹ تلاشی است برای حل این مسأله.

جواب ۱۷: مشکل این است که برنامه نویس از "و" بیتی (&) به جای "و" منطقی (&&) استفاده کرده است. "و" بیتی دو عدد به ما می دهد:

```
   3 0011
& 12 1100
=========
   0 0000
```

لذا اگر نتیجه صفر است، عبارت if اجرا نشده و قسمت else اجرا می شود. بعضی برنامه نویسان از مختصرنویسی if(x) برای if(x != 0) استفاده می کنند. این مثالی است برای اینکه چرا من از مختصرنویسی خوشم نمی آید. راه بهتر نوشتن عبارت if این است:

```
if ((i1 != 0) && (i2 != 0))
```

مدت کوتاهی بعد از اینکه من این باگ را پیدا کردم، به یکی از همکارانم گفتم. چیزی را که اتفاق افتاده بود توضیح دادم و گفتم: "من حالا تفاوت بین "و" و "وو" را می دانم".

جواب ۱۸: مسأله این است که tmp_name اشاره گری به متغیر محلی name برمی گرداند. وقتی تابع تمام می شود، تمام متغیرهای محلی غیر ایستا، از بین می روند. این قضیه شامل name هم می شود. بنابراین، اشاره گری که برگردانده شد، به مکانی تصادفی و تخصیص نیافته از حافظه اشاره می کند. فراخوانی بعدی تابع، احتمالا آن فضا را پاک می کند و a_name را کاملا عجیب و غریب می سازد. راه حل این مشکل این است که name را بصورت static تعریف کنیم. (برنامه ۵۹ مشکلی مشابه دارد).

جواب ۱۹: مشکل این است که عبارت bit >>= 1; بیت را به مکان درست خود منتقل نمی کند. در عوض، یک شیفت علامت دار انجام می دهد که بیت علامت را کپی می کند. بنابراین:

0x8000 << 1 1000 0000 0000 0000 (b) آنگونه که انتظار می رود برابر نیست با 0x4000 0100 0000 0000 (b)
بلکه برابر است با 0xC000 1100 0000 0000 0000 (b)

بخاطر همین مسأله تست بیتی، نتایج غلطی را می دهد.

جواب ۲۰: آرگومان های memset اینها هستند:

```
memset(
        void *ptr,// Pointer to the data
        int value,// Value to set
        size_t size// Number of bytes to fill
```

);

در این حالت، value برابر sizeof(array) است و تعداد بایت هایی که باید پر شود برابر ۰ است. از آنجا که size=0 است، هیچ کاری انجام نمی شود. برنامه نویس باید اینگونه می نوشت:

```
memset(array, '\0', sizeof(array));
```

جواب ۲۱: استاندارد C++ بیان می دارد که تمام اشاره گرها باید به یک آرایه یا آدرسی بالاتر از آن اشاره کنند. شما نمی توانید به آدرسی پایینتر از آرایه اشاره کنید. در این مثال، ما آرایه ای روی یک ماشین اینتل داریم. آدرس آرایه، در اشاره گر عجیب و غریب اینتل برابر 5880:0000 است. متغیر data_ptr از 5880:001E شروع می شود. سپس تا زمانیکه از data بزرگتر باشد مقدار آن کاهش می یابد. در طی کاهش یافتن مقدار آن، data_ptr به 5880:0000 می رسد. این برابر آدرس آرایه داده است، لذا دوباره شروع به کاهش می کند. (با یاد داشته باشید که در این مدل حافظه، فقط قسمت آدرس تغییر می کند). نتیجه برابر 5880:FFFE است.

حالا data =< data_prt ارزیابی می شود. ولی data_ptr اکنون خیلی بزرگتر از data است، لذا برنامه به کار خود ادامه می دهد. نتیجه این است که برنامه روی داده های تصادفی می نویسد که می تواند باعث از کار افتادن سیستم شود. ولی اگر این اتفاق نیفتد، data_ptr به 5880:0000 تنزل می یابد و این فرایند دوباره شروع می شود.

جواب ۲۲: مشکل این است که تابع max ارجاعی به یک پارامتر برمی گرداند. پارامتر برابر 3+4 است که یک عبارت می باشد. وقتی که min فراخوانی می شود، آنچه که C++ واقعا انجام می دهد این است:

۱. یک متغیر موقتی می سازد(tmp1) و مقدار 2+1 را به آن می دهد.
۲. یک متغیر موقتی می سازد (tmp2) و مقدار 3+4 را به آن می دهد.
۳. max(tmp1, tmp2) را فراخوانی می کند.
۴. تابع، یک ارجاع به tmp2 برمی گرداند.

```
i = &tmp2
tmp1 destroyed
tmp2 destroyed
```

۵. اکنون متغیر i ارجاعی است به هیچ چیز.

مشکل از برگرداندن یک ارجاع به یک پارامتر سرچشمه می گیرد. این کار باعث ایجاد یک ارجاع سرگردان می شود.

جواب ۲۳: برنامه نویس برای خروجی متن، فاصله قرار نداد:

```
13 std::cout << "Twice" << number << "is" <<
14        (number *2) << '\n';
```

در نتیجه خروجی بصورت Twice5is10 می شود. چیزی که باید می نوشت این است:

```
std::cout << "Twice " << number << " is " <<
        (number *2) << '\n';
```

(spaces added)

جواب ۲۴: این یک مسأله بن بست کلاسیک است:
پردازه ۱ به منابع #2 ,#1 نیاز دارد.
پردازه ۲ به منابع #1 ,#2 نیاز دارد.

آنها این منابع را به آن ترتیب می گیرند. به یاد داشته باشید که سوئیچ بین ریسمان ها در هرزمانی می تواند رخ دهد. بنابراین یک شرایط مسابقه داریم که ممکن است این اتفاقات در آن بیفتد:

۱. پردازه ۱، منبع 1# را بگیرد.
۲. به پردازه ۲ سوئیچ شود.
۳. پردازه ۲، منبع 2# را بگیرد.
۴. پردازه ۲ سعی کند تا منبع 1# را بگیرد.
۵. منبع 1# در دسترس نیست، لذا پردازه به خواب می رود تا زمانیکه آن منبع آزاد شود (و در این مدت، منبع 2# را قفل کرده در اختیار دارد)
۶. به پردازه ۱ سوئیچ شود.
۷. پردازه ۱ سعی کند تا منبع 2# را بگیرد. این منبع قفل است، لذا پردازه به خواب می رود تا زمانیکه آن منبع آزاد شود (و در این مدت منبع 1# را قفل کرده در اختیار دارد)

نتیجه این است که پردازه ۱ در حالیکه منبع 1# را در اختیار دارد، منتظر منبع 2# است و تا زمانیکه منبع 2# را بدست نیاورد، منبع 1# را رها نمی کند. پردازه ۲ در حالیکه منبع 2# را در اختیار دارد، منتظر منبع 1# است و تا زمانیکه منبع 1# را بدست نیاورد، منبع 2# را رها نمی کند.

ترتیب قفل شدن ها را تعریف کنید. مثلا باید به ترتیب 2# ,1# قفل کنید. همیشه وقتی چند قفل وجود دارد، این ترتیب را رعایت کنید. وقتی چند قفل دارید، از این الگوریتم استفاده کنید:

۱. سعی کنید همه قفل ها را بگیرید (اگر در دسترس نیستند، متوقف نشوید).
۲. اگر همه را گرفتید، کار خود را شروع کنید.
۳. اگر همه قفل ها را نگرفتید، آنهایی را که نگرفتید، آزاد کنید، مدت زمانی به خواب روید و دوباره شروع کنید.

جواب ۲۵: مشکل در عبارت if(n2 =! 0) است. این یک عبارت انتصاب درون یک if است. اگر برنامه را بازنویسی کنیم تا از مختصرنویسی پرهیز کرده باشیم به دو عبارت می رسیم:

```
n2 = !0;
if (n2)
```

استفاده از "نه" منطقی در این حالت (0!) به ما نتیجه ۱ را می دهد. بنابراین همیشه داریم به n2 مقدار ۱ را می دهیم، و سپس عمل مقایسه و تقسیم را انجام می دهیم. =! بصورت برعکس != نوشته شده بود و به همین دلیل این نتایج عجیب را به ما می داد. عبارت باید اینگونه باشد:

```
if (n2 != 0)
```

جواب ۲۶: مشکل این است:

```
diff[diff_index++] =
        array[i++] - array[i++];
```

این به کامپایلر می گوید که:

۱. به مقدار i، یکی اضافه کن.
۲. از آن برای اندیس آرایه استفاده کن (اولین بار).
۳. به مقدار i، یکی اضافه کن.
۴. از آن برای اندیس آرایه استفاده کن (دومین بار).
۵. تفاوت را محاسبه کن.

مسأله این است که مراحل ۱- ۴ می توانند به ترتیب دیگری اتفاق بیفتند:

۱. به مقدار i، یکی اضافه کن.
۲. به مقدار i، یکی اضافه کن.

۳. از آن برای اندیس آرایه استفاده کن (اولین بار).
۴. از آن برای اندیس آرایه استفاده کن (دومین بار).

عبارت هایی که اثرات جانبی زیادی دارند این وسعت عمل را به کامپایلر C++ می دهند که همه چیز را خراب کند.

جواب ۲۷: مشکل این است که "1" یک عدد صحیح است. عدد "3" نیز یک عدد صحیح است. بنابراین "1/3" یک تقسیم عدد صحیح است. لذا عبارت:

```
12 result = 1/3; // Assign result something
```

یک تقسیم عدد صحیح ۱ بر ۳ را انجام می دهد. تقسیم های عدد صحیح، قسمت کسری جواب را قطع می کنند بنابراین نتیجه برابر صفر است. عدد صحیح "0" به یک عدد ممیز شناور تبدیل شده و بعنوان نتیجه برگردانده می شود. برنامه نویس باید اینگونه می نوشت:

```
12 result = 1.0 / 3.0;// Assign result something
```

جواب ۲۸: استفاده از تابع scanf بسیار گول زننده است. در این برنامه عبارت:

```
22 scanf("%c %d", &oper, &value);
```

یک کاراکتر و یک عدد صحیح می گیرد. دفعه بعدی که scanf فراخوانی شود، یک کاراکتر و عدد صحیح دیگر می خواند. پس کاراکتر بعدی چیست؟ بگذارید به یک اجرای نمونه نگاه کنیم:

```
% calc
Enter operator and value:+ 5
Total: 5
Enter operator and value:+ 10
Bad operator entered
Total: 5
Enter operator and value:Bad operator entered
Total: 5
Enter operator and value:q
Bad operator entered
Total: 5
Enter operator and value:q
```

در خط اول می نویسیم: 5+
بعد از اولین فراخوانی scanf، اشاره گر ورودی درست قبل از خط جدید و درست بعد از 5 است. scanf بعدی سعی می کند تا عملگر را بخواند و یک خط جدید دریافت می کند. به خواندن ادامه می دهد و به جای یک عدد، به یک + برمی خورد. نتیجه کار بسیار گیج کننده است.

برای اینکه فریب scanf را نخورید، من یک راه به شما پیشنهاد می کنم. همیشه از ترکیبی از fgets و sscanf استفاده کنید:

```
fgets(line, sizeof(line), stdin);
sscanf(line, "%c %d", &operator, &value);
```

جواب ۲۹: پیش پردازنده، سینتکس C++ را نمی فهمد. وقتی TOTAL را به صورت 33+37 تعریف می کنیم، دقیقا و لفظ به لفظ برابر 33+37 است نه برابر 70. ماکروی AREA به صورت 10*33+37 تعریف شده است. تقدم عملگرها فرصت را غنیمت شمرده و جواب اشتباه می دهد. تا آنجا که ممکن است از ثوابت به جای ماکروها استفاده کنید. در دو طرف #define هایی که چیزی غیر از یک عدد ساده را تعریف می کنند همیشه پرانتز قرار دهید. مثال:

```
// Total top size
#define TOP_TOTAL (TOP_PART1 + TOP_PART2)
```

جواب ۳۰: مشکل این جاست که تابع، یک ارجاع به یک متغیر محلی برمی گرداند. این چیز بدی است چون با return، متغیر محلی از بین می رود و ارجاع، به یک ارجاع سرگردان تبدیل می شود. به چیزی اشاره می کند که دیگر وجود ندارد. وقتی بخواهید رشته ای را چاپ کنید که دیگر وجود ندارد، به دردسر می افتید. هیچ وقت ارجاع به متغیرهای محلی را برنگردانید.

جواب ۳۱: مسأله این است که عبارت else به نزدیکترین if تعلق دارد. کد درست توگذاری شده اینگونه است:

```
23 if (balance < 0)
24      if (balance < - (100*DOLLAR))
25              cout << "Credit " << -balance << endl;
26      else
27              cout << “Debt “ << balance << endl;
```

و این چیزی نیست که مد نظر برنامه نویس بوده باشد. چیزی که او می خواست این است:

```
if (balance < 0) {
        if (balance < - (100*DOLLAR))
                cout << "Credit " << -balance << endl;
} else
        cout << "Debt " << balance << endl;
```

از {} در دو طرف if، for، while یا هر عبارت کنترلی دیگری که بیش از یک عبارت کنترلی در خود دارد، استفاده کنید.

جواب ۳۲: مشکل این است که حافظه در تابع سازنده تخصیص یافته است و هیچ گاه آزاد نشده است. همیشه آن چه را که در تابع سازنده new کرده اید در تابع مخرب delete کنید.

جواب ۳۳: برنامه چاپ می کند:

```
First: John
Last: Smith
Hello: John
Smith
```

مشکل این جاست که fgets یک خط و از جمله newline را دریافت می کند. بنابراین وقتی اولین اسم خوانده می شود بصورت John\n است. همین اتفاق برای Smith می افتد و در نتیجه این خروجی مسخره را دریافت می کنیم.

جواب ۳۴: در انتهای عبارت for یک نقطه ویرگول اضافی وجود دارد:

```
for (index = 1; index <= 10; ++index);
```

این بدین معناست که for مطلقا کاری نمی کند. برنامه درست توگذاری شده این است:

```
for (index = 1; index <= 10; ++index);
std::cout << index << " squared " <<
        (index * index) << '\n';
```

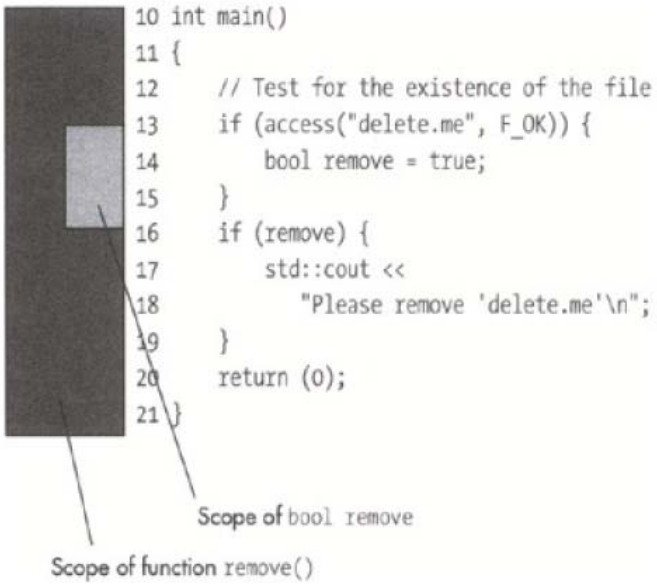
یا اگر کمی توضیح به آن اضافه کنیم، این شکلی می شود:

```
for (index = 1; index <= 10; ++index)
        /* Do nothing */;
std::cout << index << " squared " <<
        (index * index) << '\n';
```

می توانیم دریابیم که std::cout درون حلقه for نیست.

جواب ۳۵: مسأله این است که ما یک متغیر محلی به نام remove تعریف کردیم. یک تابع استاندارد به نام remove نیز وجود دارد. متغیر محلی ما، این تابع را در حوزه متغیر محلی، پوشاند. این حوزه در پایان اولین if

در خط ۱۵ به پایان رسید. عبارت بعدی یعنی if (remove) 16 بررسی می کند که آیا آدرس تابع remove غیر صفر است یا نه و اگر باشد، عبارت بعدی را اجرا می کند.



جواب ۳۶: مشکل این است که رشته ای که برمی گردانیم بدین صورت تعریف شده:

```
15 // The name we are generating
16 std::string name;
```

این یک متغیر محلی است. زیربرنامه یک ارجاع به این رشته برمی گرداند ولی از آنجا که این یک متغیر محلی است، در پایان تابع، از بین می رود. یعنی وقتی از نتیجه استفاده می کنیم، متغیری که نتیجه را در خود داشت، از بین رفته است.

جواب ۳۷: مسأله این است که کاراکتر \ بعنوان یک کاراکتر گریز[1] استفاده می شود. لذا \n یک خط جدید است. \new نیز برابر <newline>ew است. بنابراین رشته \root\new\table در واقع بدین صورت است:

```
"<return>oot<newline>ew<tab>able"
```

آنچه که برنامه نویس واقعا می خواست این بود:

```
const char name[] = "\\root\\new\\table"; // DOS path
```

این قاعده به #include اسم فایل ها اعمال نمی شود لذا #include "\usr\include\table.h" درست کار می کند.

جواب ۳۸: مشکل عبارت if (balance < 0) است. برای این استفاده شده است که آیا مشتری به کارخانه بدهکار است یا نه. بنابراین مشتری می تواند چنین پیغامی را ببیند:

```
You owe 0.
```

جواب ۳۹: مسأله این است که بهینه ساز، باهوش است. او می فهمد که ما می خواهیم نتیجه factor1*factor2; را درون حلقه for محاسبه کنیم. اگر ما این عبارت را به بیرون حلقه for انتقال دهیم، جواب فرقی نمی کند ولی همه چیز سریعتر می شود. لذا نسخه بهینه شده این برنامه، عمل ضرب را فقط یک بار انجام می دهد:

[1] Escape Character

```
17 int register1 = factor1 * factor2;
18 // We know that 1863 multiplies
19 // delay the proper amount
20 for (i = 0; i < 1863; ++i)
21 {
22        result = register1;
23 }
```

برای حل این مشکل، باید factor خود را به صورت volatile تعریف کنیم.

```
1 /*************************************************
2 * bit_delay -- Delay one bit time for *
3 * serial output. *
4 * *
5 * Note: This function is highly system *
6 * dependent. If you change the *
7 * processor or clock it will go bad. *
8 *************************************************/
9 void bit_delay(void)
10 {
11        int i; // Loop counter
12        volatile int result;// Result of the multiply
13
14        // Factors for multiplication
15        volatile int factor1 = 12;
16        volatile int factor2 = 34;
17
18        // We know that 1863 multiples delay
19        // the proper amount
20        for (i = 0; i < 1863; ++i)
21        {
22                result = factor1 * factor2;
23        }
24 }
```

چیزهایی مثل این است که برنامه نویسی تعبیه شده[1] را ساده می سازد.

جواب ۴۰: مشکل این است که ostream بصورت "ارسال با مقدار" ارسال شده است. شما نمی توانید متغیرهای stream را کپی کنید. پارامتر باید به صورت "ارسال با ارجاع" تغییر کند:

```
void print_msg_one(
        // File to write the message to
        class ostream &out_file,
        // Where to send it
        const char msg[]
)
```

جواب ۴۱: مشکل عبارت strcat(file_name, '/'); است. تابع strcat دو رشته را به عنوان آرگومان می گیرد. در این مثال، به او یک رشته و یک کاراکتر داده ایم. از آنجا که پیش الگویی وجود ندارد، C نمی تواند پارامترها را بررسی کند. پارامتر غلط به strcat ارسال می شود که بسیار گیج کننده خواهد بود.

تمام توابع باید صریحا اعلان شوند. هیچ وقت اجازه ندهید که C آنها را بطور ضمنی اعلان کند. مطمئن شوید که سرآمدها، پیش الگوی همه توابعی که شما استفاده می کنید که تعریف می کنند.

[1] Embedded

جواب ۴۲: یک عدد تک بیتی علامت دار می تواند دو مقدار داشته باشد: 0 و 1-. عبارت:
printer_status.online = 1;

درست کار نمی کند چون یک تک بیت نمی تواند مقدار ۱ داشته باشد. (لذا سرریز می کند و مقدار ۱- را به متغیر می دهد!) نتیجه این است که عبارت بعدی if (printer_status == 1) نیز درست کار نمی کند. تک بیت ها باید unsigned باشند.

جواب ۴۳: در MS-DOS چیزی مانند این را دریافت می کنید:
The answer is 4C:>#
(# is the cursor)

در یونیکس چیزی مثل این:
The answer is 4$ #

مسأله این است که برنامه نویس در انتهای عبارت std::cout، یک end of line قرار نداد. نتیجه این است که برنامه اجرا می شود، چیزی چاپ می کند و پایان می یابد درحالیکه مکان نما را در انتهای خط باقی می گذارد. سپس پردازنده فرمان اجرا شده و اعلان خود را (C:> برای MS-DOS و $ برای یونیکس) درست بعد از خروجی برنامه قرار می دهد.

چیزی که برنامه نویس باید می نوشت این است:
std::cout << "The answer is " << result << '\n';

جواب ۴۴: کاما ها می توانند برای جدا کردن عبارات C++ به کار روند. اینگونه از آنها استفاده می شود:
if (x)
    std::cout << "X set. Clearing\n", x = 0;
(لطفا این طوری برنامه ننویسید!)

عبارت one_million = 1,000,000; برابر است با:
one_million = 1,
000,
000;

یا

one_million = 1;
000;
000;

از اینجا می توانیم بفهمیم که چرا خروجی برابر ۱ است.

جواب ٤٥: مشکل این است که عبارت ch+1 یک عدد صحیح است (مقدار ۶۶). C++ این را تشخیص می دهد و تابع std::cout.operator <<(int) را فراخوانی کرده و یک عدد صحیح را در خروجی می نویسد. چیزی که برنامه نویس می بایست بنویسد این است:
std::cout << static_cast<char>(ch+1);
std::cout << static_cast<char>(ch+2);

جواب ۴۶: خروجی این است:
The double of 1 is 2
The double of 2 is 3
The double of 3 is 4
The double of 4 is 5
The double of 5 is 6

به این دلیل که DOUBLE(i+1) بسط می یابد به:

```
(i+1 * 2)
```

وقتی C++ این را می بیند، ۱ را در ۲ ضرب کرده و i را به آن اضافه می کند. این آن نتیجه ای نیست که مد نظر برنامه نویس بود. تا آنجا که ممکن است از توابع inline به جای ماکروها استفاده کنید. همیشه در دو طرف پارامترهای ماکروها از () استفاده کنید مثلا:

```
#define DOUBLE(x) ((x) * 2)
```

جواب ۴۷: عبارت if (amount = 0) مقدار ۰ را به amount می دهد، سپس نتیجه را مقایسه می کند تا ببیند که برابر صفر نیست. برابر صفر است، لذا عبارت else اجرا می شود. برنامه نویس باید می نوشت:

```
if (amount == 0)
```

جواب ۴۸: ازعبارت i = 3 – i; استفاده کنید.

جواب ۴۹: مسأله این است که تقدم عملگرهای C++ آنگونه نیست که برنامه نویس فکر می کرده است. عملگر + قبل از >> می آید لذا y = x<<2 + 1; معنی می شود به y = x << (2+1) و نتیجه برابر 4>>1 یا 8 می باشد. از قوانین ساده تقدم C++ استفاده کنید:

۱. % ,/ ,* قبل از + و – می آیند.
۲. در دو طرف هرچیزی () قرار دهید.

جواب ۵۰: چاپ می کند:

```
Hello
Hello
```

مشکل این جاست که وقتی fork انجام می شود، در بافر printf داده وجود دارد. fork دو کپی از پردازه و دو کپی از داده در بافر printf ایجاد می کند. بنابراین وقتی بعدا بافر (در دو پردازه) خالی می شود، از هر کدام از آنها یک Hello دریافت می کنیم.

جواب ۵۱: برنامه نویس هیچ وقت زحمت مقداردهی اولیه sum را بخود نداد. شما نمی توانید روی یک مقدار نامعلوم که ممکن است شامل هرچیزی باشد حساب کنید. بنابراین sum ممکن است از ۰، ۵۱۹۰، ۱۲۳، ۵ یا هر چیز دیگری شروع شود.

چیزی که برنامه نویس باید می نوشت این است:

```
9 int sum = 0;
```

جواب ۵۲: مشکل در خط زیر وجود دارد:

```
flags |= CD_SIGNAL;
```

این عملیات در برابر سوئیچ های ریسمان، محافظت شده نیست. روی یک ماشین با دستورالعمل های پیچیده، کد اسمبلی این کد اینگونه است:

```
; 80x86 assembly
orb $2,flags
```

سوئیچ های ریسمان فقط در انتهای هر دستورالعمل ممکن است رخ دهند. بنابراین این عملیات روی ماشین خانواده 80x86 دچار وقفه نمی شود. ولی روی یک ماشین RISC مانند Sparc این کد کمی فرق دارد:

```
1. sethi %hi(flags),%o0 ; Get the address of the flags in %o0,%o1
2. sethi %hi(flags),%o1
3. ld [%o1+%lo(flags)],%o2 ;%o2 = contents of the variable flags
4. or %o2,2,%o1 ;%o1 = The results of seeting the flag
5. st %o1,[%o0+%lo(flags)] ;Store results in %o0
```

اکنون، عبارت C++ وقفه پذیر است. علی الخصوص سناریوی زیر ممکن است رخ دهد:
۱. برنامه اجرا شده و دستورالعمل ۳ را کامل می کند. در این نقطه، مقدار پرچم ها در رجیستر 2o% قرار دارد.
۲. یک سوئیچ ریسمان رخ می دهد.
۳. پردازه دیگر، پرچم ها را دستکاری می کند.
۴. دوباره روی ریسمان قبلی سوئیچ می شود.
۵. مقدار قبلی پرچم ها در رجیستر 2o% قرار دارد.
۶. بیت، مقداردهی شده و نتیجه ذخیره می شود. بدلیل اینکه مقدار قبلی پرچم ها را در خود داشت، هر تغییری که در ریسمان دیگر رخ دهد، بطور تصادفی از بین می رود.

راه حل این مسأله این است که برای ممانعت از رخ دادن task switch در حین یک عبارت، از قفل ها استفاده کنیم.

جواب ۵۳: عبارت printf("%o\t", matrix[row][col]); 48 جواب را در مبنای هشت چاپ می کند. برنامه نویس اشتباهی کرده و بجای %d نوشته است %o. نتیجه این است که اعداد درست هستند فقط مبنای آنها اشتباه است.

جواب ۵۴: مسأله این جاست که 1/3 را نمی توان دقیقا بصورت ممیز شناور نشان داد. بگذارید ببینیم وقتی اعداد را بصورت دهدهی با هم جمع می کنیم چه اتفاقی می افتد.

```
1/3 = 0.33333
1/3 = 0.33333
1/3 = 0.33333
-------------
0.99999
```

بدلیل خطای گردکردن، نتیجه برابر ۱ نیست. به یاد داشته باشید که وقتی از ممیز شناور استفاده می کنید، اعداد دقیق نیستند.

جواب ۵۵: مشکل این جاست که در تابع مخرب، یک استثنا می اندازیم. وقتی برنامه به این خط می رسد:

```
if (i3 < 0)
        throw (problem("Bad data"));
```

کد استثنا، اجرا می شود و همه متغیرهای محلی را از بین می برد. این امر شامل متغیر a_stack نیز می شود. وقتی a_stack از بین برود، تابع مخرب فراخوانی می شود:

```
~stack(void) {
        if (count != 0) {
                throw (problem("Stack not empty"));
        }
}
```

تابع مخرب یک استثنا می اندازد. C++ دوست ندارد که درون یک استثنا، یک استثنا دیگر انداخته شود. وقتی این اتفاق می افتد، برنامه، تابع terminate() را فراخوانی می کند. اگر می خواهید که دومین استثنا و دیگر موارد مشابه را بگیرید، از تابع استاندارد set_terminate استفاده کنید تا از مسائل غیرمنتظره جلوگیری کرده باشید. در توابع مخرب، استثنا بیندازید.

جواب ۵۶: مسأله این جاست که تابع بازتعریف شده new بطور نادرست پیاده سازی شده است. برنامه نویس فکر می کرد که وقتی عمل new fast_bit_array انجام می شود، اندازه شیء تخصیص یافته برابر sizeof(fast_bit_array) است. وقتی که از fast_bit_arrau بعنوان یک کلاس پایه استفاده می شود، این امر

صادق نیست. در این حالت، اندازه حافظه تخصیص یافته، برابر اندازه کلاس مشتق safe_bit_array است که از fast_bit_array بزرگتر می باشد، که به اغتشاش حافظه می انجامد.

تابع عملگر new خودتان را تعریف نکنید مگر اینکه از کاری که دارید می کنید مطمئن باشید. اگر مطمئن هستید که می دانید چه کاری دارید می کنید، مطمئن شوید که واقعا واقعا مطمئن هستید. حتی در این صورت هم این کار را نکنید مگر اینکه واقعا ضروری باشد.

جواب ۵۷: مسأله این جاست که دو اعلان متغیر وجود دارد:

```
File: main.cpp
        int value = 20;
File: check.cpp
        int value = 30;
```

این بدین معناست که value برابر ۲۰ یا ۳۰ است. ولی کدام یک؟ نتیجه وابسته به کامپایلر است. اگر می خواهید که value نسبت به فایل هایی که در آنها تعریف شده است، محلی باشد، باید آنرا static تعریف کنید:

```
File: main.cpp
        static int value = 20;
File: check.cpp
        static int value = 30;
```

و کار بهتر این است که اسامی متفاوتی به آنها بدهید.

جواب ۵۸: بنا به استاندارد C++، وقتی که یک تابع عضو در کلاس مشتق، همنام با تابع کلاس پایه تعریف می کنید، تمام توابع عضو با آن نام، پنهان می شوند. بنابراین der::print_it(float) هم base::print_it(flaot) و هم base::print_it(int) را پنهان می سازد.

وقتی که print_it(2) را فراخوانی می کنید، C++ دنبال نسخه ای از print_it می گردد که بتواند از آن استفاده کند. تنها print_it قابل مشاهده، der::print_it(float) است. C++ اکنون تابعی دارد که یک int بعنوان آرگومان خود می گیرد، ولی می داند که چگونه یک int را به یک float تبدیل کند، لذا 2 را به 2.0 تبدیل کرده و از der::print_it(float) استفاده می کند.

جواب ۵۹: مشکل این جاست که ما یک تابع سازنده کپی تعریف نکرده ایم. وقتی این اتفاق می افتد، C++ یکی برای شما تعریف می کند و معمولا این کار را خوب انجام نمی دهد. تابع سازنده کپی اینگونه تعریف شده است:

```
var_array(const var_array &other) {
        data = other.data;
        size = other.size;
}
```

تابع سازنده کپی، برای ساختن یک کپی از an_array برای تابع store_it فراخوانی می شود. اشاره گر به داده، کپی می شود. وقتی var_array::~var_array در انتهای pushy فراخوانی می شود، داده را به heap برمی گرداند. وقتی در انتهای main، var_array::~var_array فراخوانی می شود، همان داده را به heap برمی گرداند. از آنجا که یک مکان حافظه را دوبار پاک می کنیم، نتیجه این می شود که یک heap خراب داریم.

همیشه به نحوی، تابع سازنده کپی را تعریف کنید. سه راه عمده این کار این می باشد:

۱. به طور ضمنی آنرا تعریف کنید.
۲. اگر می خواهید که هیچ کس قادر به فراخوانی آن نباشد، آنرا بصورت private تعریف کنید:

```
var_array (const var_array &);
// No one can copy var_arrays
```

۳. اگر از پیش فرض استفاده می کنید، آنرا در توضیح بیاورید:

```
// Copy Constructor defaults
```

بدین ترتیب، به بقیه می گویید که برنامه شما را چطور بخوانند و لذا پیش فرض C++ مشکل زا نخواهد بود.

جواب ۶۰: برنامه نویس عادت بدی دارد که فایل ها را بعد از اینکه باز کرد، نمی بندد. خیلی زود، تعداد فایل های باز به ماکزیمم رسیده و سیستم به او اجازه باز کردن فایل جدیدی را نمی دهد. در هر جا که لازم باشد باید فایل ها را بست:

```
int fd = open(cur_ent->d_name, O_RDONLY);
if (fd < 0)
        continue; // Can't get the file so try again

int magic; // The file's magic number
int read_size = read(fd, &magic, sizeof(magic));

if (read_size != sizeof(magic)) {
        close(fd); // <---- added
        continue;
}

if (magic == MAGIC) {
        close(fd); // <---- added
        return (cur_ent->d_name);
}

close(fd); // <---- added
```

برنامه نویس همچنین از opendir برای بازکردن یک دایرکتوری استفاده می کند. او هیچ وقت آنرا نمی بندد. بنابراین یک closedir لازم است.

```
void scan_dir(
        const char dir_name[] // Directory name to use
)
{
        DIR *dir_info = opendir(dir_name);
        if (dir_info == NULL)
                return;
        chdir(dir_name);
        while (1) {
                char *name = next_file(dir_info);
                if (name == NULL)
                        break;
                std::cout << "Found: " << name << '\n';
        }
        closedir(dir_info); // <---- added
}
```

جواب ۶۱: مشکل این جاست که عبارت:

```
5 const char *volatile in_port_ptr =
6 (char *)0xFFFFFFE0;
```

به C++ می گوید که اشاره گر از نوع volatile است. داده ای که به آن اشاره می شود از نوع volatile نیست. نتیجه این است که بهینه ساز، عدم وجود را برای ما بهینه سازی می کند. راه حل این است که volatile را جایی

قرار دهیم که داده مورد اشاره را دستکاری می کند. ما همچنین یک const به اعلان خود اضافه کرده ایم تا مطمئن شویم که اشاره گر قابل تغییر نیست. در نتیجه، به اعلان های زیر می رسیم:

```
4 // Input register
5 volatile char *const in_port_ptr =
6         (char *)0xFFFFFFE0;
7
8 // Output register
10 volatile char *const out_port_ptr =
11                 (char *)0xFFFFFFE1;
```

این به C++ می گوید که:

- in_port_ptr یک اشاره گر ثابت است و قابل تغییر نیست.
- int_port_ptr* از نوع volatile char است که مقدار آن را می توان ورای قوانین عادی برنامه نویسی C++ تغییر داد.

جواب ۶۲: مشکل این جاست که توضیح:

```
10 base = 5; /* Set the base of the triangle
```

بسته نمی شود. لذا عبارت زیر خود را هم بعنوان توضیح دربرمی گیرد:

```
10 base = 5; /* Set the base of the triangle
11 height = 2; /* Initialize the height */
```

از این جا براحتی می توان فهمید که چرا height مقداردهی نشده است.

جواب ۶۳: مسأله این است که getchar یک int برمی گرداند. ما داریم مقدار آنرا به یک کاراکتر می دهیم. برخی سیستم ها، کاراکترها را بعنوان کاراکترهای بدون علامت درنظر می گیرند. نتیجه این است که وقتی به EOF(-1) برسیم، سیستم این کار را می کند:

```
ch = (unsigned char)(-1)
```

یا ch=0xFF. سپس 0xFF را با 1- مقایسه کرده (که با هم برابر نیستند) و از حلقه خارج نمی شود. این برنامه یک فاجعه ادبی است. هدف هر برنامه نویس C++ باید نوشتن یک برنامه روشن و شفاف باشد. این برنامه با این هدف نوشته شد که فشرده باشد. یک برنامه خیلی بهتر می توانست اینگونه باشد:

```
1 /*************************************************
2 * copy -- Copy stdin to stdout. *
3 *************************************************/
4 #include <stdio.h>
5
6 int main()
7 {
8
9         while (1)
10        {
11                int ch; // Character to copy
12
13                ch = getchar();
14
15                if (ch == EOF)
16                        break;
17
18                putchar(ch);
19         }
20        return (0);
```

```
21 }
```

جواب ۶۴: خروجی این است:

```
Name (a): /var/tmp/tmp.2
Name (b): /var/tmp/tmp.2
```

دلیل این امر این است که با وجود اینکه ما دو اشاره گر داریم، هر دو به یک متغیر به اسم name اشاره می کنند. وقتی tmp_name برای اولین بار فراخوانی می شود:

```
a_name --> name = "/var/tmp/tmp.1"
```

بعد از دومین فراخوانی:

```
b_name --> name = "/var/tmp/tmp.2"
```

ولی a_name نیز به name اشاره می کند لذا:

```
a_name --> name = "/var/tmp/tmp.2"
b_name --> name = "/var/tmp/tmp.2"
```

دومین فراخوانی روی مکانی از حافظه که قرار بود نتیجه اولین فراخوانی را درخود داشته باشد، می نویسد. یک راه حل این است که یک رشته را بعد از هر فراخوانی کپی کنیم یا فراخواننده را مجبور کنیم که آرایه کاراکتری خود را برای ذخیره name داشته باشد. راه حل دیگر این است که از رشته های متداول C++ استفاده کنیم که خودشان تخصیص حافظه را تحت کنترل داشته باشند.

جواب ۶۵: هر put با یک flush آمده است. این بدان معنی است که برای هر کاراکتر خروجی، یک فراخوانی سیستمی انجام شده است. فراخوانی های سیستمی بسیار پرهزینه هستند و زمان زیادی از CPU را مصرف می کنند.

بعبارت دیگر، با اینکه کتابخانه I/O به منظور I/O بافرشده طراحی شده است، فراخوانی های بیش از حد flush، مانند این است که از I/O بافرنشده داریم استفاده می کنیم. ما نیاز داریم که در انتهای هر بلوک flush انجام دهیم تا مطمئن شویم که سیستم راه دور، یک بلوک کامل را دریافت می کند. یعنی بلوک، نه کاراکتر. بنابراین می توانیم با پائین بردن flush بعد از فرستادن بلوک، سیستم را تسریع بخشیم:

```
for (i = 0; i < BLOCK_SIZE; ++i) {
        int ch;
        ch = in_file.get();
        serial_out.put(ch);
}
serial_out.fflush();
```

جواب ۶۶: setjmp یک مکان را در برنامه علامت گذاری می کند. فراخوانی longjmp به آن مکان پرش می کند. مستقیما به آن جا پرش می کند، از go نمی گذرد، ۲۰۰ دلار جمع نمی کند. همچنین از همه توابع مخرب همه متغیرهای درون پشته رد می شود. در این حالت، بخاطر اینکه تابع مخرب std::string حافظه تخصیص یافته به رشته را برمی گرداند، دچار مشکل حافظه می شویم. این بدان خاطر است که توابع setjmp و longjmp، توابع C هستند که نباید در C++ استفاده شوند. از setjmp و longjmp در یک برنامه C++ استفاده نکنید. بجای آن از استثناها استفاده کنیم.

جواب ۶۷: در حالت پیش فرض:

```
defualt:
        std::cout << i << " is not prime\n";
        break;
```

کلمه کلیدی "default" درست نوشته نشده است. نتیجه این است که کامپایلر C++ فکر می کند که "defualt" یک برچسب goto است.

جواب ۶۸: تابع printf، خروجی خود را بافر می کند. در واقع، چیزی نمی نویسد تا زمانیکه بافر پر شود یا یک newline فرستاده شود. برنامه به printf می رسد، پیغام "Starting" درون بافر می رود نه به صفحه نمایش و تابع average اجرا شده و یک خطای تقسیم بر صفر دریافت می کند. نتیجه این است که پیغام "Starting" گم می شود و باعث می شود فکر کنیم که تابع average هیچ وقت فراخوانی نشده است. راه حل این مسأله این است که بعد از پیغام شروع، صریحا بافر را خالی کنیم:

```
printf("Starting....");
fflush(stdout);
```

قاعده اینکه یک بافر چه وقتی خالی می شود بسته به نوع فایلی که داریم در آن می نویسیم، فرق می کند:

۱. اگر stdout یا stderr دارند روی صفحه نمایش نوشته می شوند، خروجی بافر می شود تا زمانیکه:
   a. یک خط نوشته شود.
   b. stdin خوانده شود.
   c. بافر پر شود.

۲. اگر stdout یا stderr قرار است روی دیسک نوشته شوند، خروجی بافر می شود تا زمانیکه:
   a. بافر پر شود.

(این ها قواعدی هستند که احتمالا روی سیستم خود پیدا می کنید. قواعد واقعی، وابسته به سیستم هستند).

جواب ۶۹: مشکل این است که برنامه نویس نوشته است:

```
std::cout << "Hello World!/n";
```

بجای اینکه بنویسد:

```
std::cout << "Hello World!\n";
```

بنابراین خروجی بصورت لفظ به لفظه برابر است با:

```
Hello World/n
```

جواب ۷۰: مشکل عبارت زیر است:

```
54 while (
55        (std::strcmp(cur_cmd->cmd, cmd) != 0) &&
56                  cur_cmd != NULL)
```

این عبارت داده ای را که cur_cmd->cmd به آن اشاره می کند را بررسی می کند، سپس بررسی می کند که آیا cur_cmd->cmd معتبر است یا نه. در برخی سیستم ها، برداشتن ارجاع به NULL (که اگر در انتهای لیست باشیم این کار را می کنیم) باعث می شود که همه چیز از کار بیفتد.

در MS-DOS و دیگر سیستم های معیوب، هیچ محافظت از حافظه ای وجود ندارد، لذا برداشتن ارجاع به NULL مجاز است البته شما نتایج عجیب و غریبی خواهید گرفت. ویندوز مایکروسافت این مشکل را حل کرد . اشاره گر به NULL به یک GPF[1] می انجامد.

حلقه باید بدین گونه نوشته شود:

```
while (
```

[1] General Protection Fault

```
(cur_cmd != NULL) &&
(std::strcmp(cur_cmd->cmd, cmd) != 0))
```

ولی حتی این هم گول زننده است. این عبارت بستگی دارد به این که استاندارد C++ بدرستی پیاده سازی شده باشد. استاندارد C++ می گوید که برای && اولین قسمت ارزیابی می شود. اگر اولین عبارت غلط باشد، از عبارت دوم صرفنظر می شود. برای اطمینان بیشتر، بهتر است اینگونه بنویسیم:

```
while (1) {
        if (cur_cmd == NULL)
                break;
        if (std::strcmp(cur_cmd->cmd, cmd) == 0)
                break;
```

جواب ۷۱:

۱. تنظیم

در برخی ماشین ها باید اعداد صحیح بصورت ۲ بایتی یا ۴ بایتی باشند. در برخی ماشین ها لازم نیست. C++ برای تنظیم کردن این قاعده، از لایی گذاری استفاده می کند. لذا روی یک ماشین، ساختار به صورت زیر خواهد بود:

```
struct data {
        char flag; // 1 byte
        long int value; // 4 bytes
};
```

که کلا ۵ بایت می شود. روی یک ماشین دیگر، ممکن است اینگونه باشد:

```
struct data {
        char flag; // 1 byte
        char pad[3]; // 3 bytes (automatic padding)
        long int value; // 4 bytes
};
```

که کلا ۸ بایت می شود.

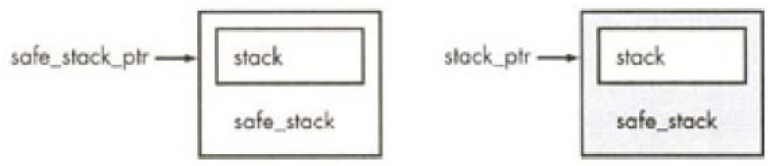
۲. ترتیب بایتها

برخی ماشین ها، اعداد صحیح بزرگ را با ترتیب بایتی ABCD می نویسند. برخی دیگر از DCBA استفاده می کنند. این امر، مانع قابلیت حمل می شود.

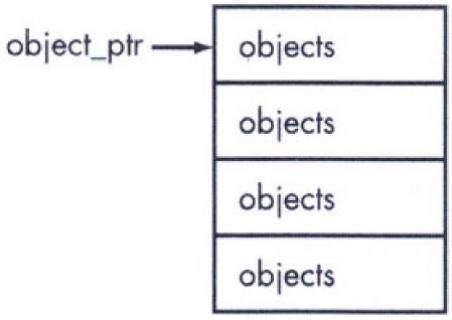
۳. اندازه عدد صحیح

ماشین های ۶۴ بیتی هم وجود دارند. یعنی روی برخی سیستم ها یک long int برابر ۶۴ بیت است نه ۳۲ بیت.

جواب ۷۲: آرایه ای از یک کلاس مشتق داریم که safe_stack نام دارد. در C++، می توانید از اشاره گری به کلاس پایه (*stack) برای اشاره به یک کلاس مشتق (safe_stack) استفاده کنید. سیستم فقط قسمت پایه کلاس را می بیند ولی با این حال می توانید به آن اشاره کنید.



حالا یک اشاره گر می تواند به یک نمونه واحد از یک کلاس یا آرایه ای از اشیاء اشاره کند.

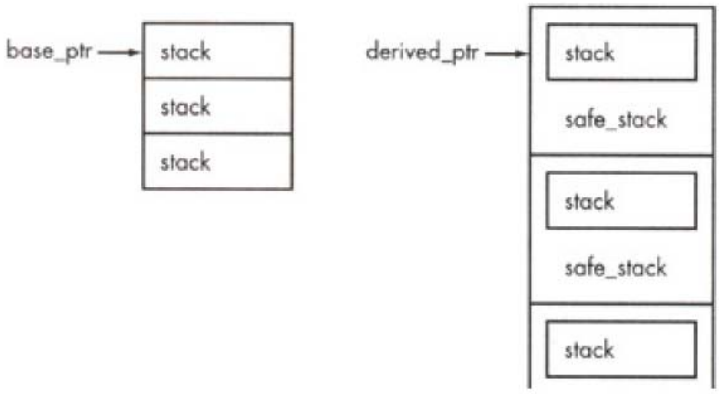


بنابراین قوانین زیر را خواهیم داشت:
۱. اشاره گر کلاس پایه می تواند به یک شیء مشتق شده اشاره کند.
۲. اشاره گر به شیء می تواند به آرایه ای از اشیاء اشاره کند.

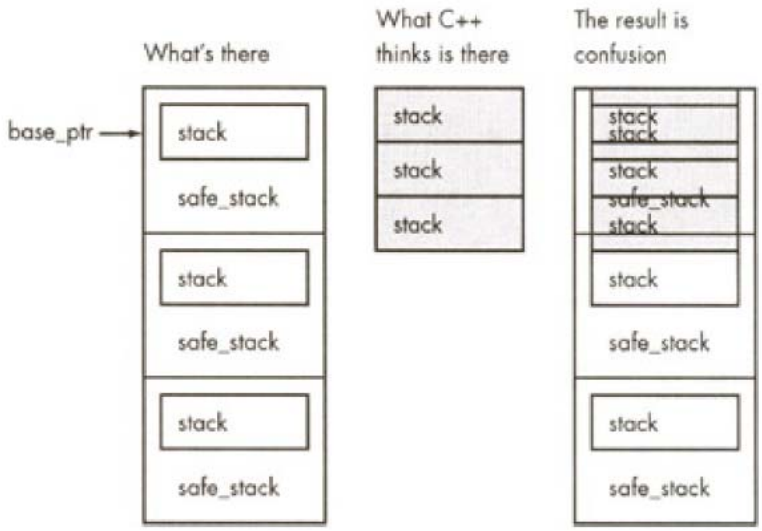
از روی این می توانیم نتیجه بگیریم:
۱. یک اشاره گر پایه می تواند به آرایه ای از اشیاء مشتق شده اشاره کند.

این غلط است.

مشکل این جاست که آرایه ای از اشیاء مشتق شده، با آرایه ای از اشیاء پایه یکسان نیست.



بنابراین اگر یک اشاره گر پایه را گرفته و آنرا به یک آرایه از مشتق شده ها اشاره دهیم، لایه حافظه غلط خواهد بود.



از قالب بردار STL بجای یک آرایه استفاده کنید. این کار کلی از مشکلات را حل می کند. آرایه های کلاس پایه را بعنوان پارامتر، ارسال نکنید.

جواب ۷۳: مسأله این است که کامپایلر چگونه کد ماشین را برای برنامه تولید می کند. عبارت:

```
if (number1 + number2 == number1)
```

چیزی مثل این را تولید می کند:

```
movefp_0, number1
add fp_0, number2
movefp_1, number1
fcmpfp_0, fp_1
jump_zero out_of_the_while
```

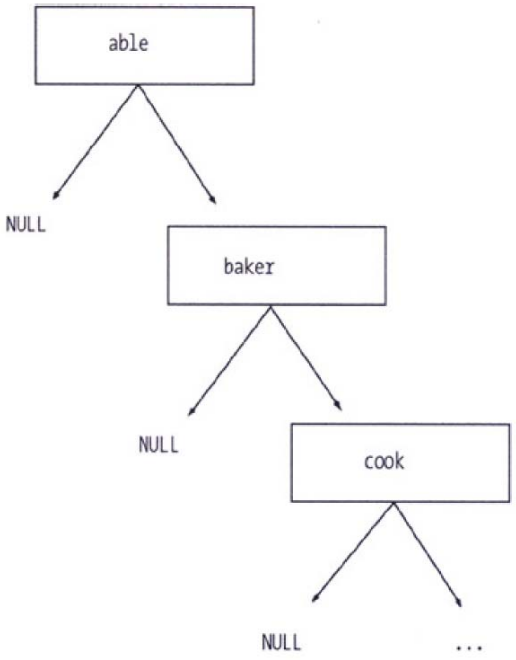
در این مثال، fp_0 و fp_1 رجیسترهای ممیز شناور هستند. در کمک پردازنده های ممیز شناور، رجیسترها، بزرگترین دقت موجود را دارند. لذا در این حالت، درحالیکه که اعداد ممکن است فقط ۳۲ بیتی باشند، پردازنده ممیز شناور، کارها را با ۸۰ بیت انجام می دهد که منجر به دقت بالایی می شود.

این نوع مشکلات در اغلب ماشین هایی که پردازنده ممیز شناور دارند رخ می دهد. از طرف دیگر، اگر یک ماشین قدیمی دارید که از نرم افزار برای انجام عملیات ممیز شناور استفاده می کنند، احتمالا جواب درستی دریافت می کنید. این بدان خاطر است که کلا، ممیز شناور نرم افزاری فقط از بیت های لازم برای انجام کار استفاده می کند.

برای درست کردن برنامه، باید حلقه اصلی را تبدیل کنیم به:

```
while (1)
{
        // Volatile keeps the optimizer from
        // putting the result in a register
        volatile float result;
        result = number1 + number2;
        if (result == number1)
                break;
```

جواب ۷۴: مشکل این جاست که کلمات به ترتیب الفبایی در فایل ورودی ذخیره شده اند و درخت، نامتوازن است. بنابراین وقتی کلمات درج می شوند، ساختمان داده زیر ساخته می شود:



نتیجه این است که یک لیست پیوندی داریم نه یک درخت. کلمات به انتهای لیست پیوندی اضافه می شوند (پرهزینه) و جستجوها بصورت خطی انجام می شود (بازهم پرهزینه). یک درخت متوازن می توانست این مشکل را حل کند.

جواب ۷۵: مسأله این جاست که در برنامه خود، این عبارت را داریم:

```
an_array = an_array;
```

این بدین صورت تغییر قیافه می دهد:

```
82 to_array = from_array;
```

تابع =operator داده های آرایه مقصد را پاک می کند. همه چیز خوب است به غیر از اینکه آرایه مبدأ نیز در همان پشته قرار دارد و داده های آن از بین می رود. راه حل این است که در تابع =operator، خود انتصابی[1] را صریحا بررسی کنیم:

```
array & operator = (const arrary &old_array) {
        if (this == &old_array)
                return;
```

تابع =operator باید خودانتصابی را بررسی کند.

جواب ۷۶: مشکل این است که strcmp مقدار صفر را در صورت برابر بودن رشته ها برمی گرداند و در غیر اینصورت مقدار غیرصفر برمی گرداند. این بدان معناست که اگر عبارت if((strcmp(x,y)) را داشته باشید، عبارت if فقط وقتی اجرا می شود که دو رشته با هم برابر نباشند. برای بررسی اینکه دو رشته باهم برابر هستند از if(strcmp(x,y) != 0) استفاده کنید. این از if(strcmp(x,y)) واضح تر است و کار می کند.

[1] Self-Assignment

تا آنجا که ممکن است از کلاس C++ string بجای رشته های قدیمی C استفاده کنید. در این صورت می توانید به جای strcmp از عملگرهای رابطه ای (<, >, ==, ...) استفاده کنید.

جواب ۷۷: مشکل در کد زیر است:

```
while (first != NULL) {
        delete first;
        first = first->next;
}
```

این داده ها را پاک کرده و سپس از آنها استفاده می کند. بعد از اینکه چیزی پاک شد، باید دور انداخته شود. همیشه بعد از delete کردن یا free کردن یک اشاره گر، مقدار آن را برابر NULL قرار دهید. وقتی برنامه با مقداری محافظت نوشته می شود، مشکل قابل مشاهده است:

```
delete first;
first = NULL;
first = first->next;
```

همچنین، بخاطر محافظت اضافه شده، یعنی مقدار دهی first به NULL، اگر بخواهیم از اشاره گر استفاده کنیم، در اغلب سیستم ها با شکست مواجه می شویم.

جواب ۷۸: نوع های متغیرها بدین صورت می باشد:

sam یک اشاره گر کاراکتری است (char *)

joe یک کاراکتر است (char)

بعد از اینکه پیش پردازنده درون اعلان می رود، به این نتیجه می رسیم:

```
char * sam, joe;
```

برای تعریف انواع جدید، به جای typedef# از define استفاده کنید.

جواب ۷۹: C++ عملگر ** ندارد (حداقل برای اعداد صحیح). بنابراین (2 ** 12) یک عبارت غلط است. مشکل این جاست که این سینتکس غلط، در یک ماکروی پیش پردازنده یک تا خط ۱۶ بسط نیافته است، پنهان می ماند. بهمین دلیل است که چرا خط ۱۶، آن خطی است که خطای سینتکس دارد. تا آنجا که ممکن است، به جای ماکروهای پیش پردازنده از const استفاده کنید. عبارت const int GROSS = (12 ** 2) نیز یک پیغام خطا ایجاد می کند ولی حداقل، شماره خط را درست می گوید.

جواب ۸۰: مشکل این است که نتیجه یک مقایسه، یک عدد صحیح برابر ۰ یا ۱ است. بنابراین عبارت if (a > b > c) تبدیل می شود به if ((a > b) > c) و چون a بزرگتر از b است، نتیجه a > b برابر ۱ می شود لذا خواهیم داشت if (1 > c) که غلط است، بنابراین عبارت else اجرا می شود.

جواب ۸۱: برنامه نویس شک دارد که چیز مسخره ای دارد رخ می دهد وقتی داده 500# در حال خوانده شدن است. او می خواهد یک نقطه انفصال[1] درست قبل از اینکه داده خوانده شود، قرار دهد. مشکل این است که اگر یک نقطه انفصال بالای get_data قرار دهد، باید ۵۰۰ دستور ادامه دیباگر را انجام دهد تا به جایی که می خواهد برسد. بنابراین نقطه انفصال را در خط ;seq = seq می گذارد.

جواب ۸۲: برنامه نویس از نقطه ویرگول برای پایان اعلان define# استفاده کرده است. از آنجا که پیش پردازنده خیلی ملالغطی است، نقطه ویرگول جزیی از متن می شود. نتیجه این است که USABLE بدین صورت تعریف می شود:

[1] Breakpoint

```
8.5; -1.0;;
```

حال، مقدار دهی اولیه text_width تبدیل می شود به:

```
double text_width = 8.5; -1.0;;
```

یا اگر درست توگذاری کنیم:

```
double text_width = 8.5;
-1.0;
;
```

حالا می توانیم مشکل خود را ببینیم. تا آنجا که ممکن است، به جای #define از const استفاده کنید.

جواب ۸۳: مسأله این است که buffer یک متغیر محلی است. این بدان معناست که در انتهای فراخوانی تابع، از بین می رود. متأسفانه، printf این را نمی داند و بنابراین همچنان درونش داده می ریزد. عبارت printf("That's all\n"); می خواهد همچنان از متغیر محلی استفاده کند. برای حل این مشکل، buffer را بصورت static تعریف کنید:

```
static char buffer[BUFSIZ];
```

جواب ۸۴: مشکل، بهینه ساز است. بهینه ساز می داند که متغیر debugging برابر صفر است. همیشه صفر است. حالا که این را می دانیم، بگذارید نگاهی به عبارت if (debugging) بیندازیم. این همیشه غلط است، چون debugging همیشه صفر است. بنابراین این بلوک هیچگاه اجرا نمی شود. یعنی می توانیم کد زیر را:

```
13 if (debugging)
14 {
15      dump_variables();
16 }
```

به این صورت بهینه سازی کنیم:

```
// Nothing
```

حال بگذارید ببینیم تعداد دفعاتی که debugging مورد استفاده قرار می گیرد، چقدر است. در خط ۱۱ مقداردهی اولیه می شود و در خط ۱۳ استفاده می شود. خط ۱۳ مورد بهینه سازی قرار گرفته است لذا debugging هیچگاه استفاده نمی شود. اگر از متغیری هیچگاه استفاده نشود، می توان آنرا با بهینه سازی حذف کرد. نتیجه، برنامه بهینه سازی شده زیر است:

```
9   void do_work()
10 {
11      // Declaration optimized out
12
13      // Block optimized out
14      //
15      //
16      // End of block that was removed
17      // Do real work
18 }
```

حالا برنامه نویس ما می خواهد از متغیر debugging برای کمک کردن در امر دیباگ استفاده کند. مشکل این جاست که بعد از بهینه سازی، متغیر debugging ای وجود ندارد. مشکل این جاست که C++ نمی دانست که برنامه نویس می خواست از جادو (یک دیباگر) برای تغییر متغیرها استفاده کند. اگر می خواهید کاری اینچنین انجام دهید، باید به کامپایلر بگویید. این کار با اعلان متغیر debugging به صورت volatile انجام پذیر است.

```
static volatile int debugging = 0;
```

کلمه کلیدی "volatile" به C++ می گوید که "چیزی عجیب و غریب مثل یک روتین وقفه، یک دستور دیباگر، یا چیز دیگری ممکن است این متغیر را تغییر دهد. نمی توانی هیچ فرضی در مورد مقدار آن داشته باشی".

جواب ۸۵: عبارت printf("The answer is %d\n"); 11 به C می گوید که یک عدد صحیح را چاپ کند، ولی آن عدد صحیح را فراهم نمی آورد. تابع printf این را نمی داند، لذا اولین عدد را از پشته بیرون می آورد (یک عدد تصادفی) و آن را چاپ می کند. چیزی که برنامه نویس می بایستی بنویسد این است:

```
printf("The answer is %d\n", answer);
```

جواب ۸۶: مشکل در استفاده از matrix[1, 2] است. عملگر کاما در C++ صرفا نتیجه قسمت دوم را برمی گرداند. بنابراین عبارت "1, 2" به C++ می گوید که قسمت اول (1) را دور بینداز و مقدار عبارت برابر 2 است. بنابراین matrix[1, 2] درواقع برابر matrix[2] است. این یک اشاره گر درون یک آرایه از اعداد صحیح است، و C++ آنرا بصورت یک اشاره گر برای چاپ کردن در نظر می گیرد. به همین دلیل است که نتایج عجیب و غریبی چاپ می شود. آنچه که واقعا مدنظر برنامه نویس بود، این است: matrix[1][2]

جواب ۸۷: نسخه پیشوندی ++ عدد را بعد از افزایش، برمی گرداند. لذا ++++i; به C++ می گوید که یکی به مقدار i اضافه کن، نتیجه را برمی گرداند و سپس متغیر i را دوباره افزایش می دهد. نسخه پسوندی ++ (i++) یک کپی از متغیر برمی گرداند، سپس مقدار آن را افزایش می دهد. بنابراین i++++:

۱. به C++ می گوید که یک کپی از i بگیر (مثل tmp_1)
۲. یکی به مقدار i اضافه کن.
۳. بقیه کارها را روی tmp_1 انجام بده.
۴. یک کپی از tmp_1 بگیر (مثلا tmp_2)
۵. یکی به مقدار tmp_2 اضافه کن.
۶. مقدار tmp_1 را بعنوان نتیجه عبارت برگردان.

از ++ و -- به صورت تکی استفاده کنید.

جواب ۸۸: مشکل در ماکروی زیر است:

```
#define SQR(x) ((x) * (x))
```

که وقتی با SQR(++number) فراخوانی می شود، بسط می یابد به

```
((++number) * (++number))
```

این عبارت، number را دوبار افزایش می دهد، به جای یک باری که مدنظر برنامه نویس بود. بدتر اینکه کامپایلر می تواند روی ترتیب اجرای عملیات مختلف تصمیم بگیرد. بنابراین، نتیجه این عبارت وابسته به کامپایلر است.

بجای ماکروهای پارامتردار از توابع inline استفاده کنید.

جواب ۸۹: بهینه ساز می داند با وجود اینکه زیربرنامه، مقدار result را محاسبه می کند، کاری با آن نمی کند. لذا چه result محاسبه شود و چه نشود، برنامه به یک صورت کار می کند. بنابراین، بهینه ساز، نگاهی به حلقه می اندازد:

```
20 for (i = 0; i < 1863; ++i)
21 {
22         result = factor1 * factor2;
23 }
```

که بهینه می شود به:

```
20 for (i = 0; i < 1863; ++i)
21 {
```

```
22        /* Do nothing */;
23 }
```

البته ما نیازی نداریم تا هیچ کار را، ۱۸۶۳ دفعه انجام دهیم لذا این هم بهینه سازی می شود:

```
20 /* No loop needed */
21 {
22        /* Do nothing */;
23 }
```

این تقریبا بهینه ترین حالت است. راه ممانعت بهینه ساز از انجام این کار این است که متغیر result را volatile اعلان کنیم. برنامه ۱۱۰ نشان می دهد که بعد از این کار چه اتفاقی می افتد.

جواب ۹۰: C++ از نمایه گذاری مبتنی بر صفر استفاده می کند. بنابراین برای array[5] عناصر معتبر عبارتند از: array[0], array[1], array[2], array[3], array[4]

با این حال، برنامه نویس از عناصر ۱ تا ۵ استفاده می کند. چیزی به اسم array[5] وجود ندارد، لذا برنامه حافظه را به طور تصادفی دستکاری می کند، که باعث برهم ریختگی حافظه می شود. به همین دلیل است که اکثر برنامه های C++ چنین عبارتی ندارند:

```
for (i = 1; i <= 5; ++i) {
```

به جای آن نوشته می شود:

```
for (i = 0; i < 5; ++i) {
```

جواب ۹۱: مسأله این است که در عبارت

```
result=result/*divisor; /* Do divide */;
```

اولین */ (آنکه در وسط عبارت قرار دارد) یک توضیح را آغاز می کند؛ یک عمل تقسیم انجام نمی دهد. لذا این عبارت معادل است با:

```
result = result /* a very big comment */;
```

در دو طرف عملگرها، فاصله قرار دهید. این کار نه تنها از مشکلات جلوگیری می کند بلکه خواندن برنامه را آسانتر می سازد.

```
result=result / *divisor; /* Do divide */;
```

جواب ۹۲: مشکل این است که یک سوئیچ ریسمان می تواند در هر زمانی رخ دهد. وقتی که count > 0 است، Writer یک کاراکتر از بافر بیرون می آورد. reader دو کار انجام می دهد:

```
++count; // We've got a new character
*in_ptr = ch;// Store the character
```

ولی یک سوئیچ ریسمان ممکن است بین این دو مرحله رخ دهد. بنابراین، این سناریو ممکن است رخ دهد:

- reader: ++count;// We've got a new character
- سوئیچ ریسمان به writer.
- writer: بررسی اینکه count > 0 – برقرار است.
- writer: کاراکتر را بگیرد.
- سوئیچ ریسمان به reader.
- reader: بعد از اینکه writer، کاراکتر را خواند، آنرا در بافر قرار دهد.

یک راه حل برای این کار این است که ترتیب مراحل:

```
++count; // We've got a new character
*in_ptr = ch;// Store the character
```

را تغییر دهیم به:

```
*in_ptr = ch;// Store the character
++count; // We've got a new character
```

اتکا به توالی دستورالعمل ها برای محافظت از داده های مشترک، مشکل و گول زننده است. بسیار بهتر و ساده تر است که به مدیر پشته بگویید که چه زمانی دارید عبارات وقفه ناپذیر را انجام می دهید. در pthreads این کار بوسیله یک قفل mutex انجام پذیر است:

```
pthread_mutex_lock(&buffer_mutex);

++count;
*in_ptr = ch;
++in_ptr;

pthread_mutex_unlock(&buffer_mutex);
```

جواب ۹۳: متغیرهای عضو به ترتیب اعلان، مقداردهی اولیه می شوند. در این صورت عبارات:

```
) : width(i_width),
        height(i_height),
        area(width*height)
```

به ترتیب اعلان اجرا می شوند. ۱) area، ۲) width، ۳) height. این بدان معناست که area با مقادیر نامعلوم width و height مقداردهی می شود و سپس width و height مقداردهی اولیه می شوند.

توابع سازنده ای بنویسید که متغیرها به ترتیبی که اعلان می شوند، مقداردهی اولیه شوند. (اگر این کار را نکنید، کامپایلر این کار را برای شما می کند و باعث درهم ریختگی می شود). هیچ گاه از یک متغیر عضو برای مقداردهی اولیه دیگر متغیرهای عضو استفاده نکنید.

جواب ۹۴: در توابع K&R، اعلان پارامتر دقیقا قبل از اولین آکولاد می آید. یعنی اعلان:

```
int sum(i1, i2, i3)
{
```

سه پارامتر از نوع پیش فرض (int) اعلان می کند. هر چیزی که بعد از آن باشد، یک متغیر محلی است. علی الخصوص،

```
int sum(i1, i2, i3)
{
        int i1; /* Local variable, not parameter */
        int i2; /* Local variable, not parameter */
        int i3; /* Local variable, not parameter */
```

نتیجه این است که به جای جمع کردن سه پارامتر، برنامه سه متغیر محلی که مقداردهی نشده اند را با هم جمع می کند. بنابراین تعجبی ندارد که چنین نتیجه عجیب و غریبی می گیریم.

جواب ۹۵: مشکل در عبارت زیر است:

```
24 sscanf(line, "%c %d", oper, value);
```

تابع sscanf بعنوان آرگومان خود، اشاره گر می گیرد. (به یاد داشته باشید که C آرگومان ها را برای داشتن نوع درست بررسی نمی کند). این برنامه برای هر کاراکتر، یک خواندن خام و یک نوشتن خام انجام می دهد. فراخوانی های سیستمی، پرهزینه هستند و این برنامه برای هر بایتی که کپی می شود، دو فراخوانی سیستمی (یک read و یک write) انجام می دهد.

برای تسریع بخشیدن به برنامه، فراخوانی های سیستمی را کاهش دهید. این کار به دو طریق انجام می شود:

۱. با استفاده از fstream های ورودی و خروجی به جای واصف های فایل، از سیستم بافرشده I/O استفاده کنید.
۲. در هر بار، بیش از یک کاراکتر را بخوانید یا بنویسید.

جواب ۹۷: مشکل در عبارت زیر است:

```
for (index = 0; string[index] != '\0'; ++index)
        /* do nothing */
return (index);
```

بعد از /* do nothing */ هیچ نقطه ویرگولی وجود ندارد. return جزئی از حلقه for است. اگر توگذاری را بطور مناسب انجام دهیم، این کد به این صورت در می آید:

```
for (index = 0; string[index] != '\0'; ++index)
        /* do nothing */
        return (index);
```

از روی این کد می توانیم ببینیم که در اولین بار، اندیس حلقه for برابر صفر است و return انجام می شود. به همین دلیل است که همه رشته ها، دارای طول صفر می باشند. چیزی که برنامه نویس می خواست این بود:

```
for (index = 0; string[index] != '\0'; ++index)
        /* do nothing */;
return (index);
```

جواب ۹۸: مسأله این است که کلاس با عملگر C++ new تخصیص نیافته است بلکه در عوض از عملگر قدیمی C malloc استفاده می کند. این کار فضای لازم برای کلاس را می سازد بدون این که تابع سازنده را فراخوانی کند. بعد برای تیر خلاص زدن، memset فراخوانی می شود تا کلاس را صفر کند.

```
result =
        (struct info *)malloc(sizeof(struct info));
memset(result, '\0', sizeof(result));
```

آنچه که برنامه نویس می بایست بنویسد، این است:

```
result = new info;
```

جواب ۹۹: عبارت out_file << ch کاراکتری به خروجی نمی فرستد. بدون توجه به اسم آن، متغیر ch از نوع عدد صحیح است. نتیجه این است که عدد صحیح در خروجی چاپ می شود. به همین دلیل است که فایل خروجی، پر از عدد صحیح است. این حالتی است که در آن، تشخیص خودکار نوع پارامترهای خروجی در C++ پا در کفش شما می گذارد. عبارت قدیمی C printf کارها را بدرستی انجام می دهد:

```
printf("%c", ch);
```

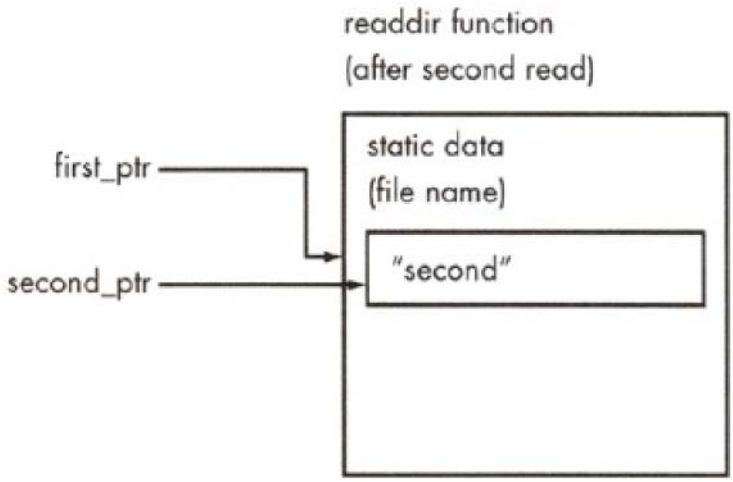
ولی با C++ باید عمل cast انجام دهید تا جواب درست بگیرید:

```
out_file << static_cast<char>(ch);
```

جواب ۱۰۰: خروجی برنامه این است:

```
First: second Second: second
```

مسأله این است که readdir اشاره گری به داده ایستا برمی گرداند. این داده متعلق به readdir است و با فراخوانی های بعدی، بازنویسی می شود. بنابراین چیزی که رخ می دهد، این است: ما scan_dir را فراخوانی می کنیم و first_ptr را به رشته first اشاره می دهیم. این چیزی است که می خواهیم، ولی آرایه ای که اسم را در خود دارد، ایستا است و وقتی readdir را دوباره فراخوانی می کنیم، از همان بافر برای ذخیره اسم second استفاده می کند. بنابراین اکنون first_ptr به second اشاره می کند که ریشه مشکلات هم همین می باشد.



جواب ۱۰۱: در تابع مخرب کلاس پایه، تابع clear را فراخوانی می کنیم. این تابع، تابع مجازی محض delete_data را فراخوانی می کند. در طول عمل تخریب، کلاس مشتق ابتدا از بین می رود. وقتی کلاس مشتق از بین رفت، تعریف delete_data هم از بین می رود. سپس تابع مخرب کلاس پایه فراخوانی می شود. در این حالت، کلاس لیست ما، بطور غیر مستقیم، delete_data را فراخوانی می کند که مجازی محض است. از آنجا که هیچ کلاس مشتقی وجود ندارد، سیستم زمان اجرا برنامه را متوقف می کند.

در توابع سازنده یا مخرب یک کلاس مجرد، توابع مجازی محض را فراخوانی نکنید.

جواب ۱۰۲: من انتظار دارم که نتایج بدین صورت باشد:

```
First 1
First 1
First 1
Second 1
Second 2
Second 3
```

ولی نتایج اینگونه اند:

```
First 0
First 0
First 0
Second 0
Second 1
Second 2
```

مشکل در عبارت return (i++); است. حالا من فهمیدم که این، یکی به i اضافه کرده و برگردانده است. مسأله این است که ++i مقدار i قبل از افزایش است. بنابراین کاری که این عبارت می کند این است:

۱. مقدار i را ذخیره کن.
۲. یکی به i اضافه کن.
۳. مقدارذخیره شده را برگردان.

لذا خطوط:

```
i = 1;
return (i++);
```

باعث می شوند که یک ۱ برگردانده شود، نه یک ۲ آنگونه که انتظار می رفت.

جواب ۱۰۳: مسأله این جاست که در برخی سیستم ها، long ها باید در یک فضای چهار- بایتی قرار داشته باشند. لذا بگذارید نگاهی به ساختار خود بیندازیم:

```
struct end_block_struct
{
        unsigned long int next_512_pos; // [0123]
        unsigned char next_8k_pos1; // [4]
        unsigned char next_8k_pos2; // [5]
        unsigned long int prev_251_pos; // [6789]
```

۶ بر ۴ بخش پذیر نیست، لذا کامپایلر دو بایت دیگر اضافه می کند تا به ۸ برسد. بنابراین چیزی که در واقع داریم این است:

```
struct end_block_struct
{
        unsigned long int next_512_pos; // [0123]
        unsigned char next_8k_pos1; // [4]
        unsigned char next_8k_pos2; // [5]
        unsigned char pad1, pad2; // [67]
        unsigned long int prev_251_pos; // [89 10 11]
```

این چیزی نیست که ما می خواستیم. عباراتی مانند assert(sizeof(end_block_struct) == 16); را در برنامه خود قرار دهید تا مراقب کامپایلرهایی که این مشکل را بوجود می آورند باشید.

جواب ۱۰۴: پیش شماره 44101 برای اعداد صحیح ۱۶ بیتی MS-DOS خیلی بزرگ است. بزرگترین مقداری که یک عدد صحیح ۱۶ بیتی می تواند داشته باشد برابر ۳۲۷۶۷ است. نتیجه این است که عدد به درون بیت علامت سرریز می کند و همه چیز اشتباه می شود.

جواب ۱۰۵: ماکروی ABORT به دو عبارت بسط می یابد. لذا نتیجه عبارت if برابر است با:

```
if (value < 0)
        std::cerr << "Illegal root" << std::endl;exit (8);
```

یا اگر بدرستی توگذاری کنیم:

```
if (value < 0)
        std::cerr << "Illegal root" << std::endl;
exit (8);
```

از این جا براحتی می توان فهمید که چرا ما همیشه خارج می شویم. به جای ماکروهای چند عبارته از توابع inline استفاده کنید:

```
inline void ABORT(const char msg[]) {
        std::cerr << msg << std::endl;
        exit(8);
}
```

اگر مجبورید از ماکروهای چندعبارته استفاده کنید، آنها را در آکولاد قرار دهید:

```
#define ABORT(msg) \
        {std::cerr << msg << std::endl;exit(8);}
```

جواب ۱۰۶: مشکل در عبارت زیر است:

```
char prev_ch = '\0';
```

از آنجا که prev_ch یک متغیر automatic است، این متغیر در ابتدای هر حلقه ایجاد و مقداردهی اولیه می شود. این بدان معناست که برای اولین if، متغیر prev_ch همیشه مقدار '\0' را در خود دارد و هیچ وقت برابر حروف دوتایی نمی شود.

جواب ۱۰۷: این برنامه اشتباه بزرگی در استفاده از ممیز شناور برای پول می کند. اعداد ممیز شناور، دقیق نیستند. ما تعداد زیادی اعداد ممیز شناور را با هم جمع می کنیم و ممکن است در این میان خطایی رخ دهد. راه حل این است که پول را بصورت دلارهای اعشاری ذخیره نکند بلکه بصورت تعداد صحیحی از سنت ها ذخیره کند. از ممیز شناور برای پول یا هر چیز دیگری که می خواهید دقیقا نمایش دهید، استفاده نکنید.

جواب ۱۰۸: فراخوانی printf، هر رشته ای را که به آن بدهید، چاپ می کند. اگر به یک رشته کاراکتری، ۱ اضافه کنید، رشته را با حذف اولین کاراکتر خواهید داشت. بنابراین:

```
printf("-xxx") prints -xxx
printf("-xxx" + 1) prints xxx
```

عبارت ((flags & 0x4) != 0) بسته به اینکه، بیت دارای مقدار باشد یا نه، ۰ یا ۱ برمی گرداند. اگر بیت دارای مقدار باشد، برنامه نویس، word- را چاپ می کند (0 + "-word"). اگر بیت، مقدار نداشته باشد، خروجی برابر word خواهد بود (1 + "-word").

جواب ۱۰۹: مشکل در تابع =operator است. این تابع اینگونه تعریف شده است:

```
trouble operator = (const trouble &i_trouble)
{
        std::cout << "= operator called\n";
        data = i_trouble.data;
        return (*this);
}
```

مقدار بازگشتی این تابع، کلاس trouble است. ولی مشکلی وجود دارد. از آنجا که تابع، هیچ ارجاعی برنمی گرداند، یک کپی از متغیر باید ایجاد شود. این بدان معناست که تابع سازنده کپی باید فراخوانی شود. این، تابع =operator را فراخوانی می کند، که عمل بازگشت را انجام می دهد و تابع سازنده را فراخوانی می کند و ...

راه حل این است که تابع =operator را مجبور کنیم که ارجاعی به کلاس برگرداند:

```
trouble& operator = (const trouble &i_trouble)
```

جواب ۱۱۰: مقدار دهی اولیه log_file می تواند new را فراخوانی کند. البته new جدید ما، از log_file استفاده می کند، لذا log_file ممکن است قبل از اینکه ساخته شود، مورد استفاده قرار گیرد که همه چیز را به هم می ریزد.

جواب ۱۱۱: مسأله این است که ترتیب مقداردهی اولیه متغیرهای سراسری، تضمین نشده است. در این حالت، a_var فرض می کند که std::cout مقداردهی اولیه شده است. ممکن است اینگونه نباشد. بگذارید بدترین حالت را درنظر بگیریم و فرض کنیم که ترتیب مقداردهی بصورت a_var, std::cout است. در این صورت، a_var ساخته می شود. تابع سازنده فراخوانی شده و یک پیغام به std::cout می فرستد. از آنجا که std::cout هنوز ساخته نشده است، همه چیز به هم می ریزد و برنامه از کار می افتد.

جواب ۱۱۲: مشکل این جاست که MAX طوری تعریف شده که لفظ به لفظ برابر متن "10=" باشد. این بدان معناست که:

```
for (counter =MAX; counter > 0; --counter)
```

بسط می یابد به:

```
for (counter ==10; counter > 0; --counter)
```

این عبارت، counter را مقداردهی اولیه نمی کند (صرفا counter را با ۱۰ مقایسه کرده و نتیجه را دور می اندازد). بدلیل اینکه counter مقداردهی اولیه نشده است، تعداد تصادفی پیغام تبریک دریافت می کنیم.

جواب ۱۱۳: فاصله ای که بعد از اسم DOUBLE وجود دارد، این ماکرو را تبدیل به یک ماکروی ساده جابجایی متن می سازد. بنابراین،

```
#define DOUBLE (value) ((value) + (value))
```

باعث می شود که DOUBLE تعویض شود با:

```
(value) ((value) + (value))
```

و این بدان معناست که خط:

```
std::cout << "Twice " << counter << " is " <<
        DOUBLE(counter) << '\n';
```

بدین صورت در می آید:

```
std::cout << "Twice " << counter << " is " <<
        (value) ((value) + (value)) (counter) << '\n';
```

(با اضافه کردن توگذاری)

راه حل: DOUBLE را بدین صورت تعریف کنید:

```
#define DOUBLE(value) ((value) + (value))
```

تا آنجا که ممکن است، به جای ماکروهای پارامتردار از توابع inline استفاده کنید. مثال:

```
inline DOUBLE(const int value) {
        return (value + value);
}
```

جواب ۱۱۴: مشکل این جاست که دست بهینه ساز در بازنویسی کد، آزاد است. برخی بهینه سازها، متغیرها را در رجیسترها قرار می دهند تا برنامه سریعتر اجرا شود. مثلا، یک نسخه بهینه سازی شده این برنامه بدین صورت است:

```
/************************************************
2 * sum -- Sum the sine of the numbers from 0 to *
3 * 0X3FFFFFFF. Actually we don't care *
4 * about the answer, all we're trying to *
5 * do is create some sort of compute *
6 * bound job so that the status_monitor *
7 * can be demonstrated. *
8 ************************************************/
9 /* --- After the optimizer --- */
10 /* --- gets through with it --- */
11 static void sum(void)
12 {
13      static double sum = 0; /* Sum so far */
14      register int reg_counter = counter;
15
16      for (reg_counter = 0;
17              reg_counter < 0x3FFFFFF; ++reg_counter)
18      {
19              sum += sin(double(reg_counter));
20       }
```

```
21      printf("Total %f\n", sum);
22      counter = reg_counter;
23      exit (0);
24 }
```

از این جا می توانیم بفهمیم که مقدار counter فقط بعد از اینکه برنامه تمام می شود، به روز می شود. اگر بخواهیم که آنرا در هر لحظه در ریسمان دیگر مورد بررسی قرار دهیم، خواهیم مرد. راه حل این است که متغیر را بصورت volatile اعلان کنیم:

```
volatile int counter;
```

آنگاه کامپایلر، هیچ فرضی در مورد اینکه از نظر بهینه سازی ، چه کاری در مورد آن می تواند بکند، ندارد و ما کدی تولید کردیم که counter را به روز نگه می دارد.

جواب ۱۱۵: من همیشه سعی می کنم مطمئن شوم که متغیر data را قبل از اینکه بازنویسی کنم، delete کنم. بنابراین مشکل حافظه نخواهم داشت. من حتی در کد زیر نیز آنرا پاک می کنم:

```
34 // Copy constructor
35 v_string(const v_string &old)
36 {
37      if (data != NULL)
38      {
39              delete[] data;
40              data = NULL;
41      }
42       data = strdup(old.data);
43 }
```

این، تابع سازنده کپی است. اولین کاری که انجام می دهد این است که ببیند آیا data چیزی درون خود دارد یا نه، و اگر داشت، آنرا delete می کند. ولی data چه چیزی می تواند در خود داشته باشد؟ ما فقط کلاس را ایجاد کردیم و آنرا مقداردهی اولیه نکرده ایم. بنابراین داریم یک اشاره گر تصادفی را پاک می کنیم و در نتیجه، برنامه از کار می افتد. تابع سازنده ای که به درستی نوشته شده باشد، این است:

```
34 // Copy constructor
35 v_string(const v_string &old):
36      data(strdup(old.data))
37 {}
```