Ray Mitchell, U99999999
MeanOldTeacher@MeanOldTeacher.com
C/C++ Programming I
Section 146359, Ray Mitchell
June 25, 2019
C1A3E0_Quiz.txt
Quiz Answers

1. D
2. B
3. C
4. C
5. E
6. B

C1A3E0 Explanations

In addition to the course book references cited below, these topics are also covered in the live lectures (in-class students) and the recorded lectures (online students).

1. **D** Note 3.2; The logical negation operator ***!*** (often pronounced "not" or "bang") produces a type **int** value of either 1 of 0 in C and a type **bool** value of either **true** or **false** in C++. 1 (or **true**) will be produced if the operand of ***!*** is non-zero or true, while 0 (or **false**) will be produced if the operand is 0 or false. In the expression *6/3 + !2.2 + 3* the sub-expression *!2.2* has a value of 0 and a data type of **int**, resulting in the entire expression having a value of 5.

2. **B** Note 3.15; Indentation is only for human convenience and is completely ignored by the compiler. The rule that determines which **if** an **else** belongs to states that an **else** always belongs to the most recent non-braced **if**. In this quiz question the first **else** belongs to the ***if*** *(6 > 5)* while the final **else** belongs to the ***if*** *(5 < 4)*.

3. **B** Note 3.17; If there is no **break**, **return**, or **goto** statement at the end of the code associated with a **case** in a **switch** statement, execution will merely continue into the code associated with the next **case**.

4. **C** Note 3.3; (ASCII) In the expression *putchar(putchar('z')- putchar('A'))* the argument of the "outer" call to *putchar* must be evaluated before that function can be called. Since *putchar* returns the underlying value of the character it prints the value of *putchar('z') - putchar('A')* is 57 decimal, which is the value of the character **9** in the ASCII character set. Thus, a **9** will always be printed last. In the expression *putchar('z') - putchar('A')* the functions may be called in either order, which will result in printing either **zA** or **Az**. Thus, either **zA9** or **Az9** will be printed.

5. **E** Notes 1.5 & 3.2; The logical AND and logical OR operators ensure a guaranteed order of operand evaluation (left-to-right) and exhibit a property known as "short-circuiting", which causes evaluation to cease as soon as the outcome is determined. In the expression *putchar('A') || putchar('\0x0')* the left function call is made first, which prints the letter **A** and returns its non-zero value. Because any non-zero value is considered to be logically true the operation short-circuits and the right operand is not evaluated. Thus, only **A** gets printed.

6. **B** Note 3.11; *25, sqrt(9.0), ++x, printf("123")* is known as a "comma" expression. The value and data type of any comma expression are the value and data type of its rightmost operand. In this case that operand is *printf("123")*, which is of data type **int**. Since *printf* returns a count of the number of characters it prints the value and data type of the entire expression are 3 and **int**, respectively.

```
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 146359, Ray Mitchell
// June 25, 2019
// C1A3E1_main.c
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function main, which attempts to compute and display
// a table of the sum of cubes from 1 through a user prompted value.
//

#include <stdio.h>
#include <stdlib.h>

//
// Calculate and display a table of the sum of cubes from 0 through a user
// prompted value.  Incorrect results will occur as the numbers get larger
// if the data type used to represent the result does not have the necessary
// range and precision.
//
// If a 16-bit signed integral type were used for variable cubicSum the first
// incorrect value would be at nbr = 19.  If a 32-bit integral type were
// used the first incorrect value would be at nbr = 304.  Using a "wider"
// integer type such as unsigned long long would increase the range somewhat
// and using type double would greatly increase the range but precision would
// be lost due to the number of digits needed.  A special math library with
// integers of virtually unlimited length would be the most accurate fix but
// would also run slower and slower as the values got larger.
//
// Algorithm description:
//    1. Get the user input value.
//    2. Initialize both the value to be cubed and the cubic sum to 0.
//    3. IF the value to be cubed is less than or equal to the user input
//       value:
//          a. Calculate the cube and add it to the cubic sum.
//          b. Display the value that was cubed and the cubic sum.
//          c. Increment the value to be cubed.
//          d. Repeat from step 3.
//
int main(void)
{
   printf("Enter a decimal integer value >= zero: ");
   int lastNbr;
   scanf("%d", &lastNbr);
   // Print table header.
   printf(
      "nbr    cubic sum\n"
      "---------------\n");
   // Loop to calculate and print each cubic sum.
   short cubicSum = 0;
   for (int nbr = 0; nbr <= lastNbr; ++nbr)
      // Print the number and calculate and print the cube sum.
      printf("%3d   %10hd\n", nbr, cubicSum += (short)(nbr * nbr * nbr));
   return EXIT_SUCCESS;
}
```

```
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 146359, Ray Mitchell
// June 25, 2019
// C1A3E2_main.cpp
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function main, which displays a user-prompted decimal
// integer value in reverse.
//

#include <iostream>
#include <cstdlib>
using std::cin;
using std::cout;

const int RADIX = 10;            // radix of number system being used

//
// Prompt the user for an integer value then print the digits of that value
// in reverse order.  If the value is negative print a minus sign last.  For
// example, an input value of -0123 would result in an output of 321- while
// an input value of 000 would result in an output of 0.
//
// Algorithm description:
//
// 1. Prompt the user for input then read it into a variable named inValue.
// 2. Display the required output message up to where the reversed value
//    should start.
// 3. Use a Boolean variable to remember if the input value was positive or
//    negative.
// 4. If the input value was negative make inValue positive.
// 5. Modulo-divide inValue by RADIX to produce its least significant
//    digit(LSD), then display that LSD.
// 6. Divide inValue by RADIX to remove its LSD and assign the result back
//    into inValue.
// 7. IF inValue is not equal to 0 repeat from step 5.
// 8. ELSE IF the original user input value was negative display a minus sign.
// 9. Finish the display.
// 10. Done!
//
int main()
{
   bool wasNegative;
   int inValue;

   cout << "Enter a decimal integer value: ";
   cin >> inValue;
   cout << '\"' << inValue << "\" in reverse is \"";
   if (wasNegative = inValue < 0)   // remember if is negative...
      inValue = -inValue;           // ...and make positive
   do                               // loop to print digits in reverse
      cout << inValue % RADIX;      // print least significant digit
   while (inValue /= RADIX);        // shift number right 1 digit & repeat
   if (wasNegative)                 // if original value was negative...
      cout << '-';                  // ...print sign
   cout << "\"\n";

```

```
62      return EXIT_SUCCESS;
63  }
```

```
1   // Ray Mitchell, U99999999
2   // MeanOldTeacher@MeanOldTeacher.com
3   // C/C++ Programming I
4   // Section 146359, Ray Mitchell
5   // June 25, 2019
6   // C1A3E3_main.cpp
7   // Windows 10 Professional
8   // Visual Studio 2019 Professional
9   //
10  // This file contains function main, which displays a user-prompted
11  // decimal integer value in words.
12  //
13
14  #include <iostream>
15  #include <cstdlib>
16  using std::cin;
17  using std::cout;
18
19  const int RADIX = 10;            // radix of number system being used
20
21  //
22  // Algorithm description:
23  //
24  // The algorithm used in the code displays a user decimal integer input value
25  // in words, one-at-a-time moving left-to-right.  If the value is negative the
26  // word "minus" is displayed first.  For example, an input value of -0123
27  // would result in a display of:
28  //   "-123" in words is "minus one two three"
29  // while an input value of 000 would result in a display of:
30  //   "0" in words is "zero"
31  // Since it uses integer division (both standard and modulo), which is not
32  // portable on older compilers if either operand is negative (Note 2.8),
33  // the input value is tested and made positive if necessary. There are no
34  // nested loops, part A is completed before part B begins, and part B is
35  // completed before part C begins.  Only one instance of the code is
36  // necessary for each part:
37  //
38  // Part A:
39  //    A1. Prompt the user, get his/her input, and output the display message
40  //        up to the point where the first word of the value is needed.
41  //    A2. If user input number is negative change it to positive and display
42  //        the word "minus", followed by a space.
43  //
44  // Part B ("for" loop is used):
45  //    Find a power of RADIX divisor that will produce the most significant
46  //    digit (MSD) of the positive number as follows:
47  //       B1. Assign 1 to a divisor variable and the positive input value to a
48  //           dividend variable.
49  //       B2. IF the value of the dividend is greater than RADIX-1:
50  //              a. Multiply the divisor by RADIX; the product becomes the new
51  //                 divisor.
52  //              b. Divide the dividend by RADIX; the quotient becomes the new
53  //                 dividend.
54  //              c. Repeat from step B2.
55  //           ELSE Proceed to Part C below.
56  //
57  // Part C ("do" loop is used):
58  //    The starting value for the divisor used in this part will be the value
59  //    computed for it in Part B above. Part C will pick off the digits of the
60  //    positive input value left-to-right and display them as words as follows:
61  //       C1. Assign the positive input value to a dividend variable.
```

```
62  //       C2. Divide the dividend by the divisor, which yields the MSD. Display
63  //           it as a word using a RADIX case switch statement.
64  //       C3. Multiply the MSD by the divisor and reduce the dividend's value
65  //           by that amount.  (This removes the dividend's MSD.)
66  //       C4. Divide the divisor by RADIX; the result becomes the new divisor.
67  //       C5. IF the new divisor is not equal to 0, repeat from step C2.
68  //           ELSE You are finished displaying the number in words!
69  //
70
71  int main()
72  {
73     cout << "Enter a decimal integer value: ";
74     int inputValue;
75     cin >> inputValue;
76     cout << '\"' << inputValue << "\" in words is \"";
77     if (inputValue < 0)                  // negative number
78     {
79        inputValue = -inputValue;         // make positive
80        cout << "minus ";                 // print "minus"
81     }
82
83     // Find a divisor that will put the number's most significant digit
84     // in the units place.
85     int divisor = 1;
86     int dividend;
87     for (dividend = inputValue; dividend > RADIX - 1; dividend /= RADIX)
88        divisor *= RADIX;                 // increase divisor
89
90     // Pick off the digits and display as English words.
91     dividend = inputValue;
92     do
93     {
94        int msd = dividend / divisor;     // current msd
95        switch (msd)                      // to print msd
96        {
97           case 0:  cout << "zero";  break;
98           case 1:  cout << "one";   break;
99           case 2:  cout << "two";   break;
100          case 3:  cout << "three"; break;
101          case 4:  cout << "four";  break;
102          case 5:  cout << "five";  break;
103          case 6:  cout << "six";   break;
104          case 7:  cout << "seven"; break;
105          case 8:  cout << "eight"; break;
106          case 9:  cout << "nine";  break;
107       }
108       dividend -= divisor * msd;        // delete msd
109       divisor /= RADIX;                 // reduce divisor
110       if (divisor)
111          cout << ' ';                   // add space between words
112    } while (divisor);                   // repeat until divisor is 0
113    cout << "\"\n";
114
115    return EXIT_SUCCESS;
116 }
```