```
Ray Mitchell, U99999999
MeanOldTeacher@MeanOldTeacher.com
C/C++ Programming I
Section 162461, Ray Mitchell
June 25, 2019
C1A3E0_Quiz.txt
Quiz Answers

1. A
2. A
3. A
4. C
5. A
6. C
```

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. **A** 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 *!(6/3+2.2)* has a value of 0 and a data type of **int**, resulting in the entire expression having a value of 3.

2. **A** 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 **else** in ***else if*** *(4 > 3)* belongs to the ***if*** *(6 > 5)* while the final **else** belongs to the ***if*** *(4 > 3)*.

3. **A** 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; For all operators except the logical AND, logical OR, conditional, and comma, the order of sub-expression (operand) evaluation is unspecified. Because of this the compiler is free to call the two functions in this quiz question in either order, and that order may change between compiles if any of the code anywhere in the program is changed.

5. **A** Note 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('B')* 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 function call on the right is then made, which prints the letter **B** and returns its non-zero value. Thus, **AB** gets printed.

6. **C** Note 3.11; *sqrt(9.0), ++x, printf("123"), 25* is known as a "comma" expression. The value and data type of any comma expression is the value and data type of its rightmost operand. In this case that operand is 25, which is of data type **int**. Thus, the value and data type of the entire expression are 25 and **int**, respectively.

```c
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 162461, Ray Mitchell
// June 25, 2019
// C1A3E1_main.c
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function main, which computes and displays a table of
// values and their 7th and 8th powers.
//

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

//
// Calculate and display a table of all integer values from 0 through lastNbr
// taken to the 1st, 7th, and 8th powers. Incorrect results will  occur as the
// values get larger if the data type used to represent the result does not have
// the necessary range.  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.
//
int main(void)
{
   int lastNbr;

   printf("Enter a decimal integer value >= zero: ");
   scanf("%d", &lastNbr);
   // Print table header.
   printf(
      " n^1          n^7            n^8\n"
      "--------------------------------\n");
   // Loop to calculate and print each power.
   for (int exp1 = 0; exp1 <= lastNbr; ++exp1)
   {
      int exp7 = exp1 * exp1 * exp1 * exp1 * exp1 * exp1 * exp1;
      int exp8 = exp7 * exp1;
      // Print the number and the powers.
      printf("%4d %12d %14d\n", exp1, exp7, exp8);
   }
   return EXIT_SUCCESS;
}
```

```cpp
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 162461, 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 hexadecimal
// integer value in reverse.
//

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

const int RADIX = 16;             // radix of number system being used

//
// Prompt the user for a hexadecimal 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 -01adc0 would result in an output of 0cda1-
// 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 a double-quote, which is the first character of the required
//    output message.
// 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 and display a minus
//    sign.
// 5. Display more of the required output message up to where the reversed value
//    should start.
// 6. Modulo-divide inValue by RADIX to produce its least significant digit
//    (LSD), then display that LSD.
// 7. Divide inValue by RADIX to remove its LSD and assign the result back into
//    inValue.
// 8. IF inValue is not equal to 0 repeat from step 6.
// 9. ELSE IF the original user input value was negative display a minus sign.
// 10. Finish the display.
// 11. Done!
//
int main()
{
   bool wasNegative;
   int inValue;

   cout << "Enter a hexadecimal integer value: ";
   cin >> hex >> inValue;
   cout << hex << '"';
   if (wasNegative = inValue < 0)   // remember if is negative...
   {
      inValue = -inValue;           // ...and make positive
      cout << '-';
   }
   cout << inValue << "\" in reverse is \"";
   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 number was negative...
      cout << '-';                     // ...print sign
   cout << "\"\n";

   return EXIT_SUCCESS;
}
```

```
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 162461, Ray Mitchell
// June 25, 2019
// C1A3E3_main.cpp
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function main, which displays a user-prompted octal
// integer value in words.
//

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

const int RADIX = 8;                // radix of number system being used

//
// Algorithm description:
//
// The algorithm used in the code displays a user octal integer input value in
// words, one-at-a-time moving left-to-right.  If the value is negative the word
// "minus" is displayed first.  For example, an input value of -0123 would
// result in a display of:
//   "-123" in words is "minus one two three"
// while an input value of 000 would result in a display of:
//   "0" in words is "zero"
// There are no nested loops, part A is completed before part B begins, and part
// B is completed before part C begins.  Only one instance of the code is
// necessary for each part:
//
// Part A:
//    A1. Prompt the user, get his/her input, and display the initial double-
//        quote of the output message.
//    A2. If the user input value is negative change it to positive and display
//        a minus sign.
//    A3. Display the positive user input value(made positive by step A2 if
//        necessary).
//    A4. Display more of the output message up to the point where the first
//        word of the value is needed.
//    A5. If the original input value was negative display the word "minus",
//        followed by a space.
//
// Part B:
//    Find a power of RADIX divisor that will produce the most significant digit
//    (MSD) of the positive number as follows:
//       B1. Assign 1 to a divisor variable and the positive input value to a
//           dividend variable.
//       B2. IF the value of the dividend is greater than RADIX-1:
//              a. Multiply the divisor by RADIX; the product becomes the new
//                 divisor.
//              b. Divide the dividend by RADIX; the quotient becomes the new
//                 dividend.
//              c. Repeat from step B2.
//           ELSE Proceed to Part C below.
//
// Part C:
```

```
//    The starting value for the divisor used in this part will be the value
//    computed for it in Part B above. Part C will pick off the digits of the
//    positive input value left-to-right and display them as words as follows:
//       C1. Assign the positive input value to a dividend variable.
//       C2. Divide the dividend by the divisor, which yields the MSD. Display
//           it as a word using a RADIX case switch statement.
//       C3. Multiply the MSD by the divisor and reduce the dividend's value
//           by that amount.  (This removes the dividend's MSD.)
//       C4. Divide the divisor by RADIX; the result becomes the new divisor.
//       C5. IF the new divisor is not equal to 0, repeat from step C2.
//           ELSE You are finished displaying the number in words!
//

int main()
{
   cout << "Enter an octal integer value: ";
   int inputValue;
   cin >> oct >> inputValue;
   cout << '"' << oct;
   bool isNegative = false;
   if (inputValue < 0)                 // negative number
   {
      isNegative = true;
      inputValue = -inputValue;        // make positive
      cout << '-';
   }
   cout << inputValue << "\" in words is \"";
   if (isNegative)                     // negative number
      cout << "minus ";                // print "minus"

   // Find a divisor that will put the number's most significant digit
   // in the units place.
   int divisor = 1;
   int dividend;
   for (dividend = inputValue; dividend > RADIX - 1; dividend /= RADIX)
      divisor *= RADIX;                // increase divisor

   // Pick off the digits and display as English words.
   dividend = inputValue;
   do
   {
      int msd = dividend / divisor;    // current msd
      switch (msd)                     // to print msd
      {
         case 0:  cout << "zero";  break;
         case 1:  cout << "one";   break;
         case 2:  cout << "two";   break;
         case 3:  cout << "three"; break;
         case 4:  cout << "four";  break;
         case 5:  cout << "five";  break;
         case 6:  cout << "six";   break;
         case 7:  cout << "seven"; break;
      }
      dividend -= divisor * msd;       // delete msd
      divisor /= RADIX;                // reduce divisor
      if (divisor)
         cout << ' ';                  // add space between words
   } while (divisor);                  // repeat until divisor is 0
   cout << "\"\n";

```

```
    return EXIT_SUCCESS;
}
```