Exercise 0 *(6 points – 1 point per question – No program required)*

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

References and 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. 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. Notes 3.13, 3.14, 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** in ***if*** *(6 > 5)*, while the final **else** belongs to the **if** in ***else if*** *(4 > 3)*.

3. 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. 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. 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.

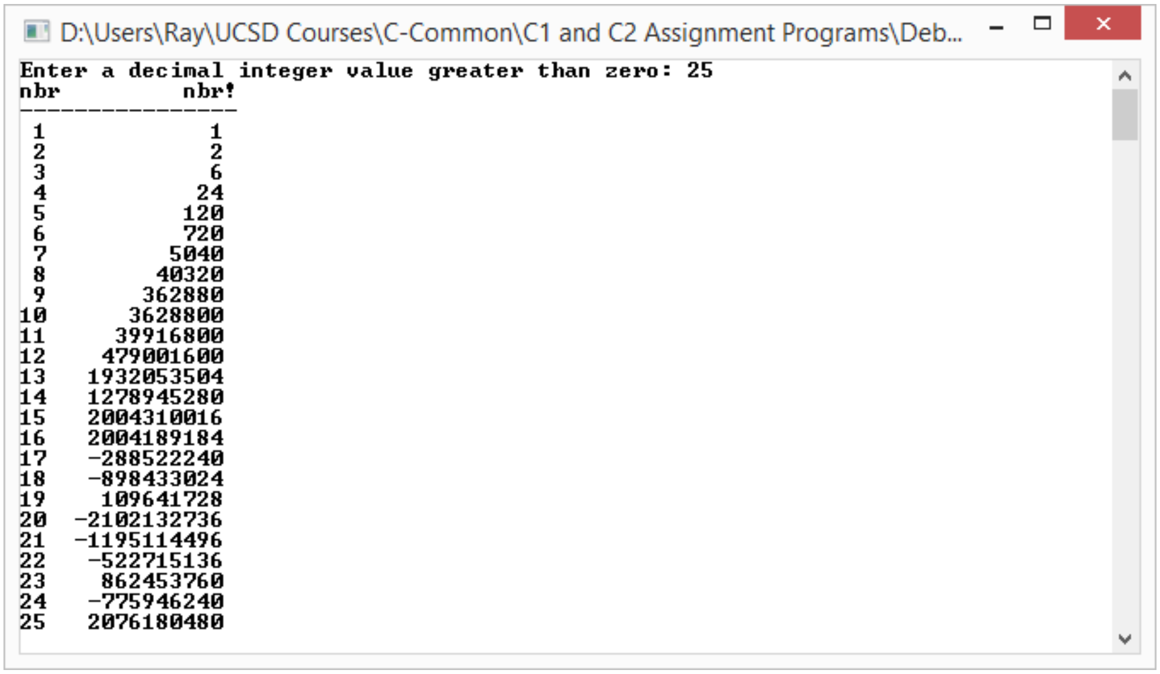
6. Note 3.11; *sqrt(9.0), ++x, printf("123"), 25* is known as a "comma" expression. The value and data type of any comma expression is the value and data type of its rightmost operand. In this case that operand is 25, which is of data type **int**. Thus, the value and data type of the entire expression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

Exercise 1 *(3 points – C Program)*

```c
/*
 * ...the usual title block Student/Course/Assignment/Compiler information goes here...
 *
 * This file contains function main, which attempts to compute and display a
 * table of factorials from 1! through a user prompted value.
 */

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

/*
 * Calculate and display a table of factorials from 1! through lastNbr!.
 * 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 integral type were used for variable factorial the first
 * incorrect value would be at nbr = 8.  If a 32-bit integral type were
 * used the first incorrect value would be at nbr = 13.  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 factorials got larger.
 *
 * Algorithm description:
 *    1. Initialize both the factorial multiplier and the factorial result
 *       to 1.
 *    2. IF the factorial multiplier is less than or equal to the desired
 *       factorial multiplier:
 *       a. Multiply factorial result by the factorial multiplier; the product
 *          becomes the new factorial result.
 *       b. Display the factorial multiplier and the factorial result.
 *       c. Increment the factorial multiplier.
 *       d. Repeat from step 2.
 */
int main(void)
{
   long factorial;
   int nbr, lastNbr;

   printf("Enter a decimal integer value greater than zero: ");
   scanf("%d", &lastNbr);
   /* Print table header. */
   printf("nbr         nbr!\n----------------\n");
   /* Loop to calculate and print each factorial. */
   for (factorial = 1L, nbr = 1; nbr <= lastNbr; ++nbr)
      /* Print the number and calculate and print its factorial. */
      printf("%2d  %11ld\n", nbr, factorial *= nbr);
   return EXIT_SUCCESS;
}
```

C1A3E1 Screen Shot is on the next page…

C1A3E1 Screen Shot (Variable *factorial* was a 32-bit **long**.)


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

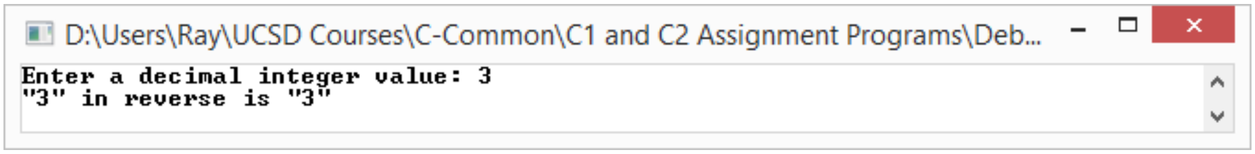
```
Enter a decimal integer value greater than zero: 25
nbr         nbr!
----------------
 1            1
 2            2
 3            6
 4           24
 5          120
 6          720
 7         5040
 8        40320
 9       362880
10      3628800
11     39916800
12    479001600
13   1932053504
14   1278945280
15   2004310016
16   2004189184
17   -288522240
18   -898433024
19    109641728
20  -2102132736
21  -1195114496
22   -522715136
23    862453760
24   -775946240
25   2076180480
```

Exercise 2 *(5 points – C++ Program)*

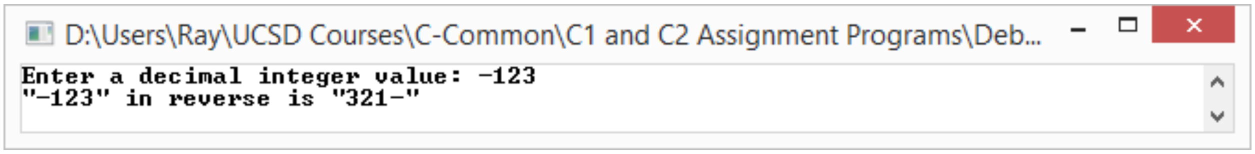
```
//
// ...the usual title block Student/Course/Assignment/Compiler information goes here...
//
// This file contains function main, which displays a user-prompted integer
// value in reverse.
//

#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:
//
// This algorithm prints the digits an input value in reverse order and prints
// a minus sign last if the value is negative. Since it uses integer division,
// which is not portable if either operand is negative, the input value must
// first be tested and changed to positive if necessary.  This positive form
// is referred to as the "number" below:
//
//    1. Modulo-divide the number by RADIX, thereby producing the least
//       significant digit (LSD), which is then displayed.
//    2. Divide the number by RADIX; the quotient becomes the new number,
//       which is the previous number with its LSD removed.
//    3. IF the new number is not equal to 0 repeat from step 1.
//    4. Display a minus sign if the original number was negative.
//
int main()
{
   bool isNegative;
   int theNumber;

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

C1A3E2 Screen Shots are on the next page...

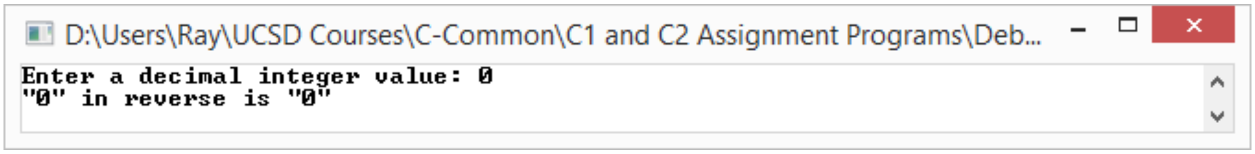
C1A3E2 Screen Shots


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

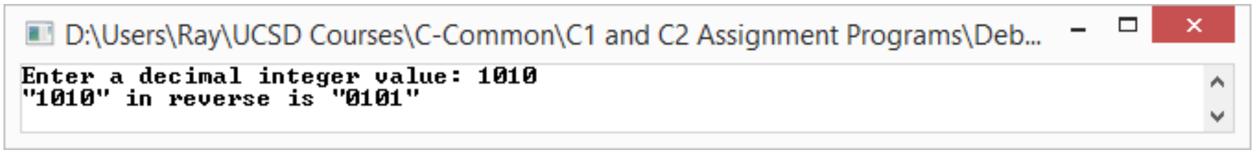
```
Enter a decimal integer value: 3
"3" in reverse is "3"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

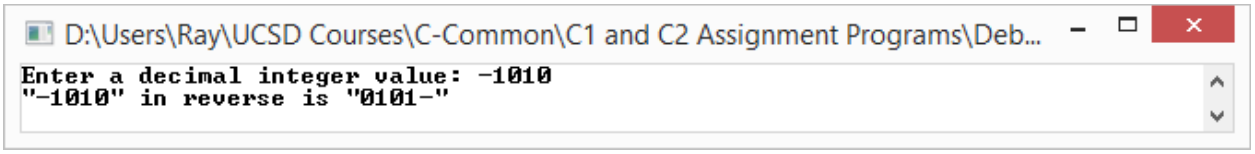
```
Enter a decimal integer value: -123
"-123" in reverse is "321-"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

```
Enter a decimal integer value: 0
"0" in reverse is "0"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

```
Enter a decimal integer value: 1010
"1010" in reverse is "0101"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

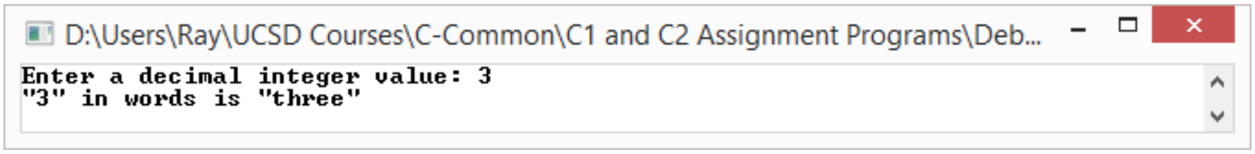
```
Enter a decimal integer value: -1010
"-1010" in reverse is "0101-"
```

Exercise 3 *(6 points – C++ Program)*

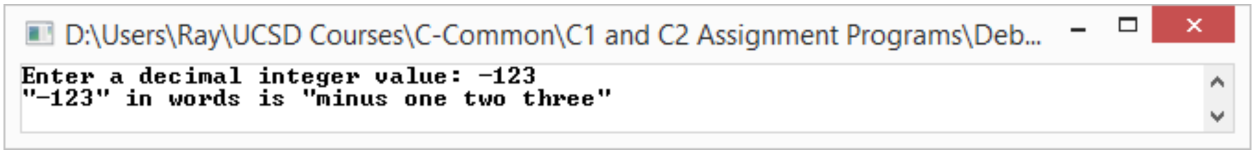
```
//
// ...the usual title block Student/Course/Assignment/Compiler information goes here...
//
// This file contains function main, which displays a user-prompted integer
// value in words.
//

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

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

//
// Algorithm description:
//
// The algorithm used in the code displays a user decimal 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"
// Since it uses integer division (both standard and modulo), which is not
// portable if either operand is negative (Note 2.8), the input value is
// tested and made positive if necessary. 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 output the display message
//        up to the point where the first word of the value is needed.
//    A2. If user input number is negative change it to positive and display
//        the word "minus", followed by a space.
//
// Part B ("for" loop is used):
//    Find a power of 10 divisor that will produce the most significant digit
//    (MSD) of the positive number as follows:
//       B1. Set a divisor variable to 1 and a dividend variable to a copy of
//           the number.
//       B2. If the value of the dividend is greater than 9:
//           a. Multiply the divisor by 10; the product becomes the new
//              divisor.
//           b. Divide the dividend by 10; the quotient becomes the new dividend.
//           c. Repeat from step B2.
//
// Part C ("do" loop is used):
//    Pick off the digits of the positive number left-to-right and display
//    them as words as follows:
//       C1. Divide the number by the divisor, which yields the MSD.  Display
//           it as a word using a 10 case switch statement (see below).
//       C2. Multiply the MSD found in the previous step by the divisor and
//           subtract the result from the number; the difference becomes the
//           new number, which is merely the previous number with its MSD
//           removed.
//       C3. Divide the divisor by 10; the result becomes the new divisor.
//       C4. If the new divisor is not equal to 0, repeat from step C1.
//

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

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

   // Pick off the digits and display as English words.
   do
   {
      int msd = theNumber / 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;
         case 8:   cout << "eight"; break;
         case 9:   cout << "nine";  break;
      }
      theNumber -= 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;
}
```

C1A3E3 Screen Shots are on the next page...

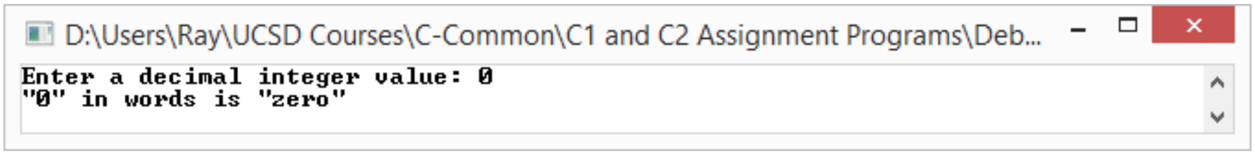
C1A3E3 Screen Shots


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

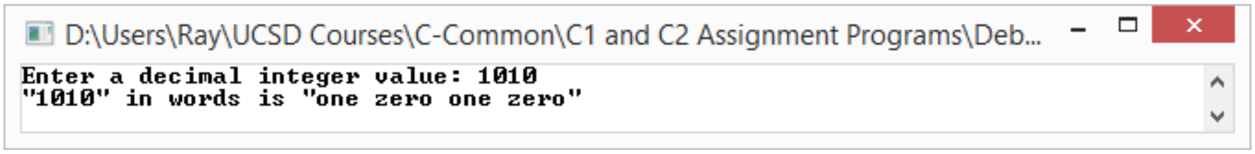
```
Enter a decimal integer value: 3
"3" in words is "three"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

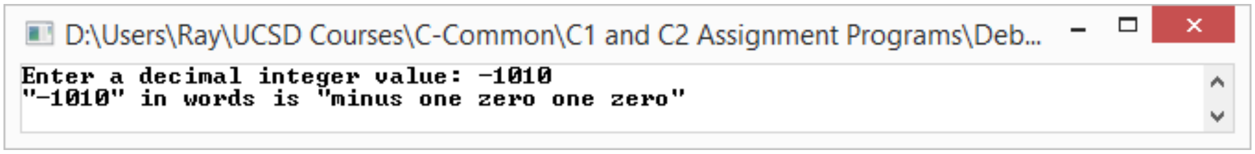
```
Enter a decimal integer value: -123
"-123" in words is "minus one two three"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

```
Enter a decimal integer value: 0
"0" in words is "zero"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

```
Enter a decimal integer value: 1010
"1010" in words is "one zero one zero"
```


D:\Users\Ray\UCSD Courses\C-Common\C1 and C2 Assignment Programs\Deb...

```
Enter a decimal integer value: -1010
"-1010" in words is "minus one zero one zero"
```