Part I: Multiple Choice (24 points)

Answer all of the following questions. READ EACH QUESTION CAREFULLY. Fill the correct bubble on your mark-sense sheet. Each correct question is worth 2 points. Choose the one BEST answer for each question. Assume that all given C code is syntactically correct unless a possibility to the contrary is suggested in the question.

Remember not to devote too much time to any single question, and good luck!

1. What are the values of x and y after executing the following programming fragment?

```
int x, y;

x = 6 + 8 / 3;
y = 42 - 7 * 3 + 1;
```

A. `x = 8, y = 22`
B. `x = 5, y = 14`
C. `x = 8, y = 20`
D. `x = 4, y = 22`
E. `x = 8, y = 140`

2. What useful operation is the following function computing?

```
double useful(double d1, double d2, double d3)
{
   if (d1 <= d2 && d1 <= d3)
      return (d1);
   else
      if (d2 <= d3)
         return (d2);
      else
         return (d3);
}
```

A. A randomly chosen one of the three given numbers
B. The sum of the three given numbers
C. The minimum of the three given numbers
D. The maximum of the three given numbers
E. The second largest of three given numbers

3. Which of the following will be the certain result of failing to fill in properly your name, student ID, section number, and exam version on your Scantron answer sheet?

A. A score of 0 will be recorded for the multiple choice portion of the final exam, regardless of how many questions you answer correctly
B. Your grade in the course will be lower than it might otherwise be since a 0 will be recorded for the multiple choice portion of the final exam
C. The grade you get for the multiple choice portion will rhyme well with the name of the Roman emperor Nero (Hint: Starts with a Z.)
D. You will need to do exceptionally well on the programming portion of this exam to help offset the 0 that you will earn for the multiple choice portion
E. All of the above

4. **For the following program:**

```
#include <stdio.h>
#include <assert.h>

int main (void) {
   int i = 2, j = 4, k = 6, scanfCount;

   printf("Enter a number: ");
   scanfCount = scanf("%d", &i);
   assert(scanfCount == 1);
   assert(i > 5);
   j = j / i;
   k = j * k;

   return 0;
}
```

Assume that when the program was executed no errors occurred (i.e., the assert statements did not fail). What was the final value of k?

A. 0
B. 1
C. 6
D. Varied, depending on the input value read into i
E. None of the above

5. **Consider the following program:**

```
#include <stdio.h>

int main(void) {
   int x = 10;
   int y = 11;

   if (x == 11)
      y = y + 3;
      x = x + 7;
   printf("x = %d, y = %d", x, y);

   return (0);
}
```

What are the values of x and y at the end?

A. x = 10, y = 11
B. x = 17, y = 11
C. x = 10, y = 14
D. x = 17, y = 14
E. The code contains a syntax error and wouldn't compile

6. **Which of the answers below is correct in light of the following code excerpt?**

```
#include <stdio.h>
#include <assert.h>

...
int newInput, scanfResult;

scanfResult = scanf("%d", &newInput);
assert(scanfResult > 1);
...
```

A. The assert statement will always pass without an error
B. The assert statement will always fail
C. The assert statement might succeed, or might fail
D. There is a syntax error
E. None of the above

7. **What are some possible reasons for defining a function?**

I. Reduce the complexity of a large section of code
II. Allow a name to be associated with a section of code for better readability
III. Make the program run faster
IV. Replace a common section of code that appears multiple times in a program

A. I and III
B. III and IV
C. I, II, and IV
D. II, III, and IV
E. II and IV

8. **Evaluate the following arithmetic expression:**

```
3.0 + 4 / 7 - (double) 4 / (5 % 3)
```

A. `1`
B. `1.0`
C. `2.0`
D. `-1.000000`
E. `-0.5`

9. Suppose we have the following prototype declaration for the function sign:

```
/* Returns 1 if n > 0, 0 if n=0, and -1 if n < 0 */
int sign(int n);
```

Which of the following functions correctly compute the absolute value of a given integer?

```
I.    int abs(int m) {
          if (sign(m) == -1) m = -m;
          return m;
      }

II.   int abs(int m) {
          return ( m * sign(m) );
      }

III.  int abs(int m) {
          if (sign(m) == -1) return (-m);
      }
```

A. I only
B. I and II
C. I and III
D. I, II and III
E. None of the above

10. In the following code fragment:

```
#include <stdio.h>

...
int i = 10, j = 30, k = 20;
double x = 0.0;

scanf("%d", &i);
if (i >= 10) {
   j = 5;
   x = (double)j/i;
}
if (x > 1)
   k = 1;
else
   k = x/2;
```

What is the final value of k?

A. 0
B. 1
C. 10
D. 20
E. It varies, depending on the value of i

11. **Which of the following are NOT valid variable names in C?**
    I. pink_floyd
    II. C++
    III. U2
    IV. 2nd_edition

    A. I and IV
    B. II and III
    C. II, III and IV
    D. II and IV
    E. None of the above

12. **What is the output of the following program when it is executed?**

```
#include <stdio.h>

#define M 3

double f(void) {
   return (M);
}

double g(void) {
   double a;
   a = f() + 2;
   printf("%f   ", a);

   return (a);
}

int main(void) {
   double x;

   x = g() - f();
   printf("%.2f", x);

   return (0);
}
```

A. `3.00   1`
B. `5.000000   2.00`
C. `3   2`
D. `2   5.000000`
E. `3.0   3.14159   1.00`

Part II: Programming Questions (24 points)

Write C code for the following two problems.

13. (10 points) **Write a program which reads in an integer value, then verifies that the input operation succeeded and that the value is non-negative (Hint: Use assert statements for doing the verification), and outputs 0 if the number is even and 1 if it is odd. DO NOT use "if"-statements.**
Note: You do not have to use all the space left between the comments, if you don't need it.

```
#include <stdio.h>
#include <assert.h>

int main(void) {

    /* Declare all necessary variables here */









    /* Read in the input value and use assert statements to verify  */
    /* that the process finished successfully and that the value is */
    /* non-negative                                                  */











    /* Compute and print the necessary output */









    return (0);
}
```

14. (14 points) **Write a program, which calculates shipping and handling costs for packages based on their weight. Follow closely the outlined directions:**

1. **Print a message to the user requesting the input weight.**
2. **Read in the weight (in pounds) of a pending shipment – a value of type double. Verify that the input operation finished successfully and that the input weight is not negative.**
3. **Compute the shipping and handling charge as follows. For each 10 pounds worth of weight there is a flat charge of $3.50, for any additional weight (of less than 10 pounds) the charge is $0.40 per pound, where the additional weight is rounded to the nearest pound downwards (e.g. 3.7 pounds is considered 3 pounds).**
   Example: 43.7 pounds = 4 * 10 pounds + 3.7 pounds; charge = 4 * $3.50 + 3 * $0.40 = $15.20
4. **Output the computed shipping and handling charge.**

```
#include <stdio.h>
#include <assert.h>

/* Put appropriate #define statements for useful constants here */




int main(void) {
   /* Declare all necessary variables here */




   /* Print a message requesting the input weight, then read in   */
   /* the value and verify (use assert statements) that the input */
   /* process finished successfully and the weight is non-negative*/




   /* Compute the shipping and handling price based on      */
   /* the input weight, according to the instructions above */




   /* Inform the user of what the shipping and handling charge is */


   return (0);
}
```