Part I: Multiple Choice (22 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 is the value of **y** that is printed by the following program?

```
#include <stdio.h>
int fun1(int a) {
  return (-a);
}
void fun2(int a) {
  a = -a;
}
int main(void) {
   int x = 24, y;
   y = fun1(x);
   fun2(y);
   printf("%d", y);
   return (0);
}
A.
   0
B. 24
C.
   -24
   -1
D.
E.
   Undefined
```

2. Which of the following operations can be performed on *entire* arrays in C?

- A. Copying one to another with an assignment statement
- **B.** Passing them as parameters
- **C.** Initializing them to zero
- **D.** Comparing two of them with a relational operator
- E. None of the above

3. Given the following code:

```
int scanfcount;
int inputvalue;
scanfcount = scanf("%d", &inputvalue);
```

How would you check that the input succeeded and that the number read is a positive number divisible by 7, and quit the program if this did not happen?

```
A. assert((scanfcount == 1) && (inputvalue != 7));
B. assert(((inputvalue / 7) != 0) && (inputvalue >= 0));
C. assert(((inputvalue / 7) != 0) || (scanfcount == 1));
D. assert((scanfcount == 1) && ((inputvalue % 7) == 0) && (inputvalue >= 0));
E. assert((inputvalue > 0) && ((inputvalue % 7) != 0));
```

4. Consider the following variable declarations:

```
int i;
char s[10], c;
double *dp;
double d;
```

Assuming that all variables have been initialized previously, which of the following assignment statements are legal?

dp = &d;I. *dp = d;II. III. i = &dp;IV. &d = dp;V. s[3] = c; A. I, II, IV and V **B.** II and V C. I, II and V **D.** III and IV E. None of the above

5. Consider the following declaration and initialization of the variable Arr:

int Arr[] = {23, 1, 2, 3, 4, 13, 5};

What are the values of Arr[2] and Arr[7] respectively?

```
A. 2 and 5B. 1 and 5
```

C. 3 and 0

- **D.** 2 and undefined
- **E.** 2 and 0

6. Consider the following fragment:

```
#include <stdio.h>
...
int x = 6;
if (x > 7)
    if (x < 9) printf("8");
    else printf("not 8");
...</pre>
```

What is printed when this code fragment is executed (if anything at all)?

A. not 8B. 8

B. 8

- **C.** "8" **D.** "not 8"
- **E.** Nothing is printed

7. How many times does the line "I love CSE 142!" get printed when executing the following program fragment?

```
#include <stdio.h>
int main(void)
{
   int i, k, j = 5, m = 2;
   for (i = 0; i <= j; I++) {
      k = 0;
      while (k \le m) {
         printf("I love CSE 142!\n");
         j = j - 1;
         k = k + 1;
      }
   }
}
A.
   12
B.
   4
С.
   6
D.
   15
Е.
   18
```

8. What output is produced when the following program is executed?

```
#include <stdio.h>
void printval(int *p, int q)
{
   q++;
   if (*p != q)
     printf("%d %d ", *p, q);
   *p = *p + 1;
   if (*p != q)
      printf("%d %d ", *p, q);
   q++;
}
int main(void)
{
   int j = 7;
   printval(&j, j);
   j++;
   printf("%d", j+2);
}
   7
A.
      8
        10
B. 8
      7
         9
C. 8
      7 10
D.
   7
      8
         11
E.
   7
      8
         9
```

- 9. What are some reasons to use arrays in programming?
 - I. Operating on infinitely many variables at the same time
 - II. Using loops effectively
 - III. Storing large amounts of data
 - IV. Storing a collection of data of the same type under a common name
 - A. I and II
 - **B.** II, III and IV
 - C. I and III
 - D. III and IV
 - E. I, III and IV

10. Consider the following function definition:

```
int func(int Arr[], int n)
{
    int i, s = 0;
    for (i=0; i<n && Arr[i]>=0; i++)
        s = s + Arr[i];
    if (i!=0)
        return ((double)s / i);
    return (0);
}
```

Assuming func is called with an array of length n as its first argument, what operation does the function perform?

- A. Computes the median of all n elements in the array Arr
- B. Computes the average of all positive elements in the array Arr
- **C.** Computes the average of a sequence of elements from the array Arr up until (but not including) the first negative value
- **D.** Computes the median of a sequence of elements from the array Arr up until (and including) the first negative value
- E. Executes incorrectly due to a 'subscript out of range' error
- **11.** In C, what mechanisms are available to allow a called function to transmit data back to the caller?
 - I. scanf
 - II. return value
 - III. reference parameters
 - IV. array subscript
 - A. I only
 - B. I and III
 - C. I and IV
 - **D.** II and III
 - E. I, II and IV

Part II: Programming Questions (22 points)

```
12. Consider the following program:
```

```
#include <stdio.h>
/* function prototypes */
void func1(int *first, int *second);
int func2(int *first);
/* main function */
int main(void)
{
   int num1 = 11;
   int num2 = 12;
   func1(&num1, &num2);
   printf("num1 = %d, num2 = %d", num1, num2);
   return (0);
}
void func1(int *first, int *second)
{
   if (*first > *second) {
       *first = func2(first);
       *second = func2(first);
    }
   else {
       *first = func2(second);
       *second = func2(second);
    }
}
int func2(int *first)
{
    int count;
   for (count = *first; count > 0; count--)
        *first = *first + 1;
                                              /* <---- */
   return (*first);
}
```

a) (2 points) What output does that program produce?

b) (8 points) Draw a diagram (like the examples presented in class), that shows the condition of the program when execution has reached the return statement in func2 (marked by <----) the *first* time that func2 is executed. Your diagram should include a box for each active function that contains local variables and parameters for that function. Show the values for each parameter and local variable. If a parameter is a pointer, show the relationship between that pointer and the variable it refers to by drawing an arrow.

13. (12 points) Write a program that reads a sequence of *positive* integer values n, where the end of the input is indicated by -1. For each positive number n read, print a random number between 1 and n. If a non-positive number n different from -1 is read, skip it and read the next number. In case of input errors (i.e. non-numeric input entered), quit the program immediately.

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>
int main(void)
{
    /* Define necessary variables here */
```

```
/* Initializing the random number generator */
srand( (unsigned)time( NULL ) );
```

```
/* Put your code here */
```

return (0);

}