Part I: Multiple Choice (30 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 output from the following program?

```
#include <stdio.h>

int main(void)
{
   int i, j=1;
   for (i=3; i>0; i=i-1)
      j = j + 5.0 * i;
   printf("%d ", j);
}
```

A. 16 26 31
B. 16 26
C. 26
D. 31
E. 31 26 16

2. Consider the following list:

2 3 15 16 20 22 24 25 35

Suppose that you represented this list as an array of integers and used binary search to determine if 21 were located in it. What elements of this list would be accessed for a comparison, and in what order from first to last?

A. 16, 20, 22
B. 20, 22
C. 20, 24, 22
D. 2, 3, 15, 16, 20, 22
E. None of the above

3. Which of the following statements about *good programming practices* **when using loops is TRUE?**

A. You *need to* use doubly-nested for-loops to step through the elements of a one-dimensional array.
B. *Never* use a for-loop unless you know the exact total number of iterations *at compile time*.
C. *Always* update the loop control variable as the first statement inside a while-loop.
D. *Anything* that can be done with a for-loop can also be done with an appropriately written while-loop.
E. All of the above.

4. If you perform a binary search on a sorted array of size N, how many comparisons total will you have to do in the worst case? Express the answer as a function of N.

A. N * $\log_2(N)$
B. N
C. N/2
D. $\log_2(N)$
E. N * N

5. What possible problems do you see in the following expression,

```
!flag || (x=1 && y<3.5 && ch==C)
```

where the variables `flag` and `x` are integers, `y` is a double, `ch` is a character and `C` is not a variable?

I. The programmer may have wanted to use `x==1` instead of `x=1`.
II. The comparison `ch==C` does not properly check whether the variable `ch` contains the character `C`.
III. The above is syntactically incorrect and would not compile anyway.
IV. The expression `!flag` is incorrect, since `flag` is not being compared to anything.

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

6. What is the purpose of the following function?

```
#include <string.h>

int fn1(char ar1[], char ar2[])
{
   int i=0;

   if (strcmp(ar1, ar2) != 0) {
      while (ar1[i]==ar2[i] && i<strlen(ar1) && i<strlen(ar2))
         i++;
      return i;
   }
   else
      return (strlen(ar1));
}
```

A. It counts the number of characters in `ar1`
B. It returns the length of `ar1`
C. It counts the number of identical characters at the beginning of `ar1` and `ar2`
D. It counts the number of times the word in `ar1` appears in `ar2`
E. It swaps the letters in ar1 with those of `ar2`

7. Consider the following definitions:

```
typedef struct {
   int x, y;
} point;

/* set the x and y coordinates of the point parameter to 0 */
void destination(point *p) {
   /* sequence of statements */
}

int main(void) {
   point line[2];
   destination(&line[1]);
   return (0);
}
```

What sequence of statements should be in the body of the function `destination` **if we need to set the** `x` **and** `y` **values of** `line[1]` **to** `0`**?**

A. `p.x = 0;  p.y = 0;`
B. `p[1].x = 0; p[1].y = 0;`
C. `*(p.x) = 0; *(p.y) = 0;`
D. `p->x = 0;  p->y = 0;`
E. `&p->x = 0;  &p->y = 0;`

8. What is the output of the following program?

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

int main(void) {
   char word[10] = "CSE142";
   int length;

   length = strlen(word);
   printf("length = %d", length);
}
```

A. `length = 6`
B. `length = 7`
C. `length = 10`
D. `length = 11`
E. None of the above

9. Given the following code:

```
typedef struct{
   int x, y;
} Point;

typedef struct{
   int green, red, blue;
} Color;

typedef struct{
   Color color;
   Point point;
} ColoredPoint;

...
int i;
ColoredPoint point[50];
...
/* Assume there is code here to initialize both */
/* the array point and the integer i */
...
point[i].point = squid(point[i].point.x);
```

What is the type of the parameter passed to squid?

A. array of `ColoredPoint`'s
B. array of `Point`'s
C. a single `ColoredPoint`
D. a single `Point`
E. `int`

10. Which of the following is NOT true of structs?

A. Structs can be used to group together values of different types.
B. Structs can be used as elements of arrays.
C. Structs can be compared using operators >, <, >=, <=, and ==.
D. Structs can be returned from functions.
E. Two or more of the above statements are false.

11. Suppose we have the following line of code in a program:

```
beer[2].type = 3;
```

Which statement best describes `beer`?

A. `beer` is an integer.
B. `beer` is a struct.
C. `beer` is an array of integers.
D. `beer` is an array of structs.
E. `beer` is a 2-dimensional array.

12. Given the following code:

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

char first[]  = "Hello";
char second[] = "World";
int size;
...
size = strlen(first);
if (size == 5) {
   if (strcmp(first, second) == 0)
      printf("%s %s", first, second);
   else
      strcpy(second, first);
   printf("%s", second);
}
printf("%s", first);
```

What is printed out?

A. Hello World World Hello
B. Hello
C. World
D. Hello Hello
E. World World

13. Which of the following logical expressions is equivalent to

```
(x >= 35) && !(y < 7)
```

A. !((x >= 35) || !(y > 7))
B. !((x < 35) || (y < 7))
C. !((x < 35) && (y < 7))
D. !((x < 35) || (y >= 7))
E. !((x < 35) && (y >= 7))

14. What parameters need to be passed to a function that opens an external file for use in a C program?

I. The filename
II. Whether the file is being opened for input or output
III. The data to be written into the file

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

15. What is the output of the following program?

```
#include <stdio.h>

int f (int x) {
   if (x==0) return (0);
   if (x==1) return (1);
   else
      return (f(x-2) + x);
}

int main(void) {
   int result;

   result = f(4);
   printf("%d", result);

   return (0);
}
```

A. 0
B. 1
C. 3
D. 6
E. None of the above

Part II: Programming Questions (30 points)

16. (10 points) **Consider the following definitions:**

```
/* college years */
#define FRESHMAN  0
#define SOPHOMORE 1
#define JUNIOR    2
#define SENIOR    3

typedef struct {
   int    sid;       /* student identification number  */
   char   name[50];  /* name of student                */
   int    year;      /* college year of student        */
   double gpa;       /* grade point average of student */
} Student;
```

Write a function, `printHonorRoll`**, which accepts an array of** `Student` **and prints out in descending order of college year, the student identification number and name of every student who has a GPA greater than or equal to a specified value.**
*Hints***: You do** *not* **need to sort the array in order to print the requested output. You also** *should not* **have to write four almost identical sequences of statements, each of which prints the honor roll for one college year.**

```
/* Given an array student with nstudents records, print the ID  */
/* number and name of each student whose gpa is greater than or */
/* equal to gpa. The names and ID numbers should be sorted by   */
/* year, with seniors being printed first.                      */
void printHonorRoll(Student student[], int nstudents, double gpa)
{









}
```

17. (13 points)
(a) (5 points) **Write a function** `int appearsIn(char ch, char str[])`**, which returns 1 (true) if** `ch` **appears in the C string** `str`**, and 0 (false) otherwise.**

```
int appearsIn(char ch, char str[])
{




}
```

(b) (8 points) **Write a function** `int sameLetters(char str1[], char str2[])`**, which takes two strings as parameters and returns 1 if they contain the same set of letters, and 0 otherwise. Two strings that contain the same letters do** *not* **have to have these letters in the same order or in the same number. You** *must* **use the** `appearsIn` **function from part** (a) **in your solution to part** (b) **if you need to determine whether a particular character appears in a string.**
Example: **"steal" and "stall"** *do not* **have the same set of letters, since the first word contains the letter 'e', which is not part of the second word;**
Example: **"salt" and "stall"** *do* **have the same set of letters, since all letters in "salt" are in "stall" and vice versa.**

```
int sameLetters(char str1[], char str2[])
{




}
```

18. (17 points) **Imagine writing a program that plays chess. One thing that needs to be done is to discover whether the king is in check, i.e. if its current location is threatened by some opponent's piece. Your task is to write a set of functions, that answers this question.**

For this problem we will use a 2-D array to represent the contents of the chess board. For each cell on the board, we store the kind and color of the chess piece on that square. Here are the declarations for appropriate constants, types, and variables.

```
#define BOARD_SIZE 8 /* number of rows and columns on the chess board*/
#define EMPTY      0 /* representation of an empty board cell        */
#define KING     100 /* representation of a cell containing the king */
...                  /* similar constants for other chess pieces     */

/* colors (as in GP142) */
#define BLACK      0
#define WHITE      1

typedef struct {   /* definition of one chess board cell       */
   int pieceKind;  /* kind of piece in this cell or EMPTY      */
   int pieceColor; /* color of this piece if cell is not EMPTY */
} cell;

/* The chess board */
cell board[BOARD_SIZE][BOARD_SIZE];
```

(a) (8 points) **Write a function** `find_piece` **that searches for a specified piece of a given color and reports where it appears on the chess board if it is found or else reports that the piece is not found.**

```
/* Search for the specified pieceKind with the specified pieceColor  */
/* on the board. If the piece is found, set row and col to its       */
/* location and return true (1). If it is not found, return false (0)*/
/* and leave row and col unchanged.                                  */
int find_piece(cell board[BOARD_SIZE][BOARD_SIZE],
               int pieceKind, int pieceColor, int *row, int *col)
{









}
```

(b) (9 points) **Write a function** `is_check`**, which reports whether the king is in check. You should assume that the following function has already been written and that you can call it to determine if a cell on the board is threatened.**

```
/* Returns true (1) if the square on the given row and column is */
/* threatened, and false (0) otherwise.                          */
int is_threatened(cell board[BOARD_SIZE][BOARD_SIZE],
                  int row, int col);
```

You should use the functions `is_threatened` **and** `find_piece` **from part** (a) **as needed in your solution. If** `find_piece` **cannot find the king, print a suitable error message.**

```
/* Returns true (1) if the king of the given color is in check */
/* (i.e. the square it currently occupies is threatened).      */
/* If the king is not threatened, return false (0).            */
int is_check(cell board[BOARD_SIZE][BOARD_SIZE], int kingColor)
{









}
```