EE 285 Name_____sec_____
Exam 1 – Oct. 12, 2016

Problem 1 (25 points) — Write the expected output of the program at each of the "printf" steps below. Write comments at various points to describe the program — this helps you understand what the program is doing and helps the grader understand what you are doing.

```
//EE 285 - Exam 1, problem 1
// Oct. 12, 2016

#include <stdio.h>

int main( void ){

    int i, j, temp, minValue = 10, minLoc = 0;

    int a[5] = {7, 4, 8, 5, 2};

    for(j = 0; j < 5; j++){     // Print out the array here.
        printf("%d ", a[j]);
    }
    printf("\n\n");

/************************************/

    for( i = 0; i < 5; i++){
        if (a[i] < minValue){
            minValue = a[i];
            minLoc = i;
        }
    }

    if (minLoc != 0){
        temp = a[0];
        a[0] = a[minLoc];
        a[minLoc] = temp;
    }

    for(j = 0; j < 5; j++){     // Print out the array here.
        printf("%d ", a[j]);
    }
    printf("\n\n");

/************************************/

    minValue = 10;
    minLoc = 1;

    for( i = 1; i < 5; i++){
        if (a[i] < minValue){
            minValue = a[i];
            minLoc = i;
        }
    }

    if (minLoc != 1){
```

```
        temp = a[1];
        a[1] = a[minLoc];
        a[minLoc] = temp;
    }

    for(j = 0; j < 5; j++){     // Print out the array here.
        printf("%d ", a[j]);
    }
    printf("\n\n");

/*************************************/

    minValue = 10;
    minLoc = 2;

    for( i = 2; i < 5; i++){
        if (a[i] < minValue){
            minValue = a[i];
            minLoc = i;
        }
    }

    if (minLoc != 2){
        temp = a[2];
        a[2] = a[minLoc];
        a[minLoc] = temp;
    }

    for(j = 0; j < 5; j++){     // Print out the array here.
        printf("%d ", a[j]);
    }
    printf("\n\n");

/*************************************/

    if( a[3] > a[4]){
        temp = a[3];
        a[3] = a[4];
        a[4] = temp;
    }

    for(j = 0; j < 5; j++){     // Print out the array one last time.
        printf("%d ", a[j]);
    }
    printf("\n\n");

    return 0;
}
```

Extra credit (10 points). The above program is not written very efficiently. Use another loop to re-write the program to accomplish the same thing, but with (many) fewer lines of code.

Problem 2 (25 points) — Write a C program that does the following:

1. Declares a two-dimensional array of integers, 4 rows x 4 columns.
2. Declares a one-dimensional array of doubles, having 4 elements.
3. Ask the user to enter integer values for each of the 16 elements of the 2D array.
4. Uses a function to calculate the average value of each *row* of the array. The computed average is returned as a double. The returned values are stored in the elements of the one-dimensional array.
5. Finally, the elements of one-dimensional array are sorted in ascending order — lowest first, highest last. (Use the bubble sort method.)
6. The elements of the one-dimensional array of average values are printed in the sorted order.

You can include whatever other variables are needed to do the job.

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

EE 285 exam "help sheet"

integer — `int anInteger;`
floating point — `double theDooble;`
character — `char c;` —> character literals are denoted with single quotes. For example: `'g'`, `'T'`.

linear array — `int linArray[20];`
2D array — `double bigArray[10][5]` (10 rows, 5 columns)

`%` —> Modulus operator. Returns the remainder of a division. Example: `23%4` = 3.

```
printf( "int = %d and floating point = %lf", anInteger, theDooble );
```

`scanf( "%d", &anInteger );` —> reads an integer from the keyboard.
`scanf( "%lf", &theDooble);` —> reads an double length floating point from the keyboard.

`>=` —> greater than or equal to `( topValue >= maxValue )`
`<=` —> less than or equal to `( myAge <= 100 )`
`==` —> is equal to? `( x == 16)` double equals signs are an easily overlooked error.

`&&` —> logical AND, `( a > 0 && a <= 7)`

`||` —> logical OR, `( a >= MAX_VALUE || c == 'Q')`

`i++` —> increment by 1. Equivalent to `i = i + 1`. Equivalent to `++i`.
`i--` —> decrement by 1. Equivalent to `i = i - 1`. Equivalent to `--i` .

`a += delta` —> add `delta` to variable `a`. (Equivalent to `a = a + delta`.)
Same for `-, *, /`.

```
if( conditional statement ) {

    Do some stuff;
}
else {

    Instead do the other stuff;
}
```

The `else` part is optional (Note: if only one statement follows the if or else, then the braces are optional.)

```
while( conditional statement ){

     Do some stuff;
}

for( i = MIN; i <= MAX; i++) {

     Do some stuff;
}
```

(Again, if only one statement follows the `while` or `for`, then the braces are optional.) With the while loop, make sure that something is changing within the loop so that the conditional will become false at some point. (Avoid infinite loops.)

`if( a == 1){`
`  printf("%d: Get ready.", a);`
`}`
`else if (a == 2){`
`  printf("%d: Get set.", a);`
`}`
`else if (a == 3){`
`  printf("%d: Go!", a);`
`}`
	`switch( a ){`
	
	`  case 1:`
	`  printf("%d: Get ready.", a);`
	`  break;`
	
	`  case 2:`
	`  printf("%d: Get set.", a);`
	`  break;`
	
	`  case 3:`
	`  printf("%d: Go!", a);`
	`  break;`
	`}`

Function declaration

Note: depending on how you have ordered the functions in your program file, you may need to include a prototype declaration before using the function. Here is an example of a function definition.

```
int bigger( int a, int b){

    if( a >= b){
        printf( "%d is bigger!", a );
        return a;
    }
    else {
        printf( "No! %d is bigger!", b );
        return b;
    }
}
```