ENEE 140, Spring 2014
Final Exam

Tuesday, May 20, 2014, 1:30 – 3:30 pm

University of Maryland Honor Pledge

The University is committed to Academic Integrity, and has a nationally recognized Honor Code, administered by the Student Honor Council. In an effort to affirm a community of trust, the Student Honor Council proposed and the University Senate approved an Honor Pledge. The University of Maryland Honor Pledge Reads:

> "I pledge on my honor that I have not given or received any unauthorized assistance on this examination (or assignment)"

Please write the exact wording of the Pledge, followed by your signature, in the space below:

Pledge: ____________________

Pledge: ____________________

Pledge: ____________________

Pledge: ____________________

Your signature: ____________________

Full Name: ____________________ Section: __________ Directory ID: ____________________

1 (16):	
2 (6):	
3 (5):	
4 (15):	
5 (10):	
6 (10):	
7 (18):	
8 (20):	
TOTAL (100):	

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name, your section and your Directory ID on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 100 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. Calculators are allowed, but no other electronic devices. Good luck!

Problem 1. (16 points)

This problem tests your understanding of C types and casts. Assume that variables a, b, c and d are defined as followed:

```
int         a = 1;
unsigned    b = 2;
float       c = 1;
float       d = 2;
```

Fill in all the empty cells in the table below. For each of the C assignment expressions in the left column, state the resulting value of the r2 – r9 variables.

Assignment	Value
`float r1 = c/d;`	0.5
`int   r2 = a/d;`	
`float r3 = a/d;`	
`int   r4 = (a + 0.2) / b;`	
`int   r5 = r1 > (float)r2;`	
`int   r6 = (a - b) > 0;`	
`float r7 = ((float)a + b) * 2 / (c + d);`	
`int   r8 = (int)a + (int)c - (int)b - (int)d;`	
`int   r9 = 2.0 * a / (c - d);`	

Problem 2. (6 points)

This problem tests your understanding of variable scope in C. What is the output of the two programs below?

```
// Program 1

#include <stdio.h>

int a = 1;

int
a_plus_one()
{
      return a+1;
}

int
main()
{
      a = 2;
      printf("%d\n", a_plus_one());
      return 0;
}
```
	```
	// Program 2

	#include <stdio.h>

	int a = 1;

	int
	a_plus_one()
	{
	      return a+1;
	}

	int
	main()
	{
	      int a = 2;
	      printf("%d\n", a_plus_one());
	      return 0;
	}
	```

Output of Program 1: ___________
Output of Program 2: ___________

Problem 3. (5 points)

This problem tests your understanding of random number generation in C. Write a statement that will generate a random double between [1,2] inclusive. You may assume that `srand()` has already been invoked.

```
double x = ____________________________________________________________;
```

Problem 4. (15 points)

This problem tests your understanding of C loops. Given an integer `array A[10]`, write a snippet of code using a loop to reverse the order. Write your answer in the box on the next page.

Note that you must use a loop; an answer such as

```
      A[0] = A[9]
      A[1] = A[8]
      ...
```

will NOT be accepted.

Problem 5. (10 points)

This problem tests your understanding of C command line arguments. First, fill in the blanks to make the program print every command line argument. Then, on the next page, write the output of the program when launched as follows:

```
      a.out ENEE 140 is the best!
```

```
int
main (int argc, char *argv[])
{
      int x;

      printf("argc=%i\n",argc);
      for (x=___; x ___ ___; x++) {
            printf("argv[%i]=%s\n",x,argv[x]);
      }

      return 0;
}
```

Program output:

____

____

____

____

____

____

____

____

Problem 6. (10 points)

This problem tests your understanding of C multidimensional arrays. Fill in the blanks to correctly create a 2D array filled with 0's and 1's alternating with 8 rows and 8 columns. In other words, the content of the 2D should be:

```
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
```

```
int A[___][___];
int i,j;

for (i = ___; i ___ 8; ___) {
     for (j = ___; j ___ 8; ___){
          if (__________){
               A[i][j] = 1;
          } else {
               A[i][j] = ___;
          }
     }
}
```

Problem 7. (18 points)

This problem tests your understanding of file I/O in C. The following program should copy the file "readme.txt" into file "writeme.txt" character-by-character. Find all the bugs in the implementation.

HINT: There are 6 bugs.

```
#include <stdio.h>

int
main()
{
   FILE *file1, file2;
   char c;

   file1 = fopen ("readme.txt", "w");
   file2 = fopen ("writeme.txt", "r");

   while ((c = getc(file1)) == EOF) {
      fprintf (file1, "%c", c);
   }

   fclose (file1);
   fclose (file2);

   return 0;
}
```

Problem 8. (20 points)

This problem tests your understanding of C arrays and of sorting operations. The following function inserts a new element into a sorted array in such a way that the array remains sorted after this operation. The function returns the position where the value was inserted. Fill in the blanks below to implement this functionality:

```
// Insert a value in a sorted array. Assume that the array is
// large enough to store one more element.
//
// Parameters:
//          val                 value to be inserted
//          num_sorted          number of sorted elements in the array
//          a                   array with sorted elements
//
// Return:
//          position where the element was inserted
int
insert_sorted(int val, int num_sorted, int a[])
{
      int i;

      for (i = num_sorted; i _____ 0; i--) {
            // Invariant: val >= all elements of a[] seen so far

            if (a[i-1] > val)
                  a[_____] = a[i-1];     // shift element one position to the right
            else
                  _____;
      }

      a[i] = val;

      return _____;
}
```