University of Maryland College Park

Dept of Computer Science

CMSC106 Fall 2012

Midterm III

Last Name (PRINT): _____

First Name (PRINT): _____

University Directory ID (**e.g., umcpturtle**)_____

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

Instructions

- **Make sure you write your name now (we will not wait for you at the end).**
- This exam is a closed-book and closed-notes exam.
- Total point value is 100 points.
- The exam is a 50 minutes exam.
- Please use a pencil to complete the exam.
- WRITE NEATLY.
- You don't need to use meaningful variable names; however, we expect good indentation.

Grader Use Only

#1	Problem 1 (Miscellaneous)	(14)	
#2	Problem 2 (Memory Map)	(16)	
#3	Problem 3 (Arrays)	(30)	
#4	Problem 4 (Strings)	(20)	
#5	Problem 5 (Arrays)	(20)	
Total	Total (100)	(100)	

Problem #1 (14 pts)

1. (2 pts) Is the following assignment valid? Briefly explain.

```
int b[3];
b = NULL;
```

2. (2 pts) Would the following code compile? If not, indicate why.

```
int main() {
   int a[] = {10, 20, 40};
   int *p;

   p = a + a;
   printf("%d\n", *p);

   return 0;
}
```

3. (2 pts) Is every character array a string? Briefly explain.

4. (2 pts) Does a "b" (double quotes) occupy more space in memory than 'b' (single quotes)? Briefly explain.

5. (2 pts) Is every string a character array? Briefly explain.

6. (2 pts) Complete the following directive so we can use the string library.

```
#include <
```

7. (2 pts) Which values are returned by the **strcmp** function?

Problem #2 (16 pts)

Draw a memory map to the right of the following program that shows the values of variables when execution has reached the point indicated by /* HERE */.

```
#define LENGTH 4

void process(int m[]) {
   int *k = m;
   int w;

   m += 2;
   *m += 1000;
   m++;
   *m += 3000;
   w = m - k;
   m = NULL;
   /* HERE */
}

int main() {
   int c[LENGTH] = {2, 6, 7, 10};
   int d[LENGTH] = {0};
   int *q, idx = 0;

   q = c;
   while (*q != 10) {
      d[idx] = *q;
      idx++;
      q = q + 1;
   }

   process(d);

   return 0;
}
```

Problem #3 (30 pts)

Implement a function called **rotate_left** that rotates an array **n** times to the left. Elements will appear on the right side of the array as they are rotated. For example, rotating [10, 3, 7] 2 times generates [7, 10, 3].

```
void rotate_left(int *array, int n, int array_length)
```

Problem #4 (20 pts)

Implement a function called **create_official_name** that takes a first name and a last name, and generates an official name by creating a string with the last name, a comma, and the first name. The function will return -1 if either the first or last name (or both) are NULL, and 1 otherwise. You may not use loops in order to implement this function. If you use loops you will lose significant credit. Below we have provided an example that relies on this function.

```
char official[81];
printf("%d\n", create_official_name("Tom", "Smith", official));
printf("%s\n", official);
```

Output

```
1
Smith,Tom
```

```
int create_official_name(const char *first_name, const char *last_name,
                         char *official_name)
```

Problem #5 (20 pts)

Implement a function called **same_elements** that returns 1 if two arrays have the same number of elements and the same elements (no matter the order) and 0 otherwise. Below we present examples of the results we expect for different arrays.

[10, 3, 7] and [3, 7, 10] → 1
[10, 3, 7] and [10, 3] → 0
[10, 3, 7] and [3, 7] → 0
[10, 3, 7] and [7, 3, 10] → 1

```
int same_elements(int *a, int a_length, int *b, int b_length)
```

ADDITIONAL PAGE IN CASE YOU NEED IT