CSC 230-002 Exam 2   Spring 2014

Name: ____

Unity ID: ____

The written portion of the exam is worth 105 points, but a 100 is a perfect score. For the written portion of the exam, you may have one 8.5 by 11 inch “cheat sheet” of paper (front and back) with notes on it with you for the exam. You may NOT use any other materials, including your text book, classnotes, cell phone, neighbor, etc. You may not share your “cheat sheet”.

If you are caught violating these rules by the teaching staff, you will receive a -100 on the exam and will have an academic integrity violation filed against you.

Sign the honor pledge below to affirm that you followed the rules of this exam period. Your exam will not be graded if you do not sign the honor pledge.

“***I have neither given nor received unauthorized aid on this test or assignment”***

Signature:____

Date: ____

#	Topic	Total Points	Awarded Points
1	Preprocessor Macros	13	
2	Scope	12	
3	Storage Class	10	
4	Functions with Pointers	10	
5	Pointer Arithmetic	12	
6	String Functions	14	
7	Using Pointers	14	
8	Dynamic Memory Allocation	20	
	TOTAL	105	

1. Preprocessor Macros *(13 points)*
Write a C preprocessor macro called DEBUG that prints the name and value of a variable. For example, the following program

```
int foo = 5;
DEBUG(foo);
DEBUG(foo+1);
```

should have the following output:

```
    ‘foo’ is 5
    ‘foo+1’ is 6
```

This macro should only be defined if TESTING is equal to 1, and should keep security in mind. You will lose points for syntax errors.

2. Scope *(12 points)*
Read the code below and specify the output of the program.

```
int foo(int x)
{
        static int a;
        int b = x;
        x += 1;
        printf("%d %d %d\n",a,b,x);
        a++;
        return b++;
}

int a;
int x;
void main()
{
        a = 10;
        int b = foo(a);
        printf("%d %d %d\n",a,b,x);
        foo(a);
}
```

What is the output? There will be three lines of three integers each.

3. Storage Class (10 points)

Using the code from Question 2, determine how the memory is allocated and the storage class for each given variable at the given line number.

Variable	Line Number	Memory Allocation	Storage Class
a	3		
b	4		
a	11		
x	12		
b	16		

4. Functions with Pointers *(10 points)*
What does the function f, below, do? Frame your response as the javadoc-style comments you would provide for the function - Say what its purpose is, not what each statement or operator does. What are the inputs, outputs, and side effects? What does the output mean?

```
/**
 * How would you document this function?
 */
char *f (char *s, char c)
{
      while ( *s && *s++ != c )
      {
            // do nothing
      }

      if (*s)
      {
            return *s;
      }
      else
      {
            return NULL;
      }
}
```

5. Pointer Arithmetic *(12 points)*
The grid below represents the memory starting at address 0x100, the same location that array `A` starts at. Each element is one byte. *The grid is purposely not aligned to match the columns and rows of A.*

A							

Read the following code, and for each of the pointers (p,q,r,s,t,u), please write in the grid above which memory location is being pointed to by that pointer. Each one is worth 2 points - show your work for partial credit.

```
char A[2][5][3]; // starts at 0x100

a.)   char *p = A[2][3];
b.)   char **q = A[1]+1;
c.)   char *r = &(A[0][1][-2]);
d.)   char **s = q+1;
e.)   char *t = r+4;
f.)   char *u = *(A[0]+1)+1;
```

6. String Functions *(14 points)*
The strcmp function compares two string and returns a number equal to, less than, or greater than zero depending on the lexicographic ordering of the two strings. A string is greater than another string if the first character between the two that differs is greater. For example:

> "**b**ob" is greater than "**a**lice"
> "a**r**row" is greater than "a**p**ple"
> "bob**b**y" is greater than "bob" (the null byte is less than all other characters)

Write the strcmp function yourself, without using any string functions from the standard library. The man page for strcmp is reproduced below.

```
SYNOPSIS
      int strcmp(const char *s1, const char *s2);

DESCRIPTION
      The  strcmp()  function compares the two strings s1 and s2.  It returns
      an integer less than, equal to, or greater than zero if  s1  is  found,
      respectively, to be less than, to match, or be greater than s2.

RETURN VALUE
      The strcmp() function returns an integer less than, equal to, or
      greater than zero if s1 is found, respectively, to be less than,
      to match, or be greater than s2.
```

8. Using Pointers *(14 points)*
In the last homework assignment, you had to develop a stack, which is a first-in, first-out data structure. For this question, we want you to use an array to develop a *queue* of integers *without using the array indexing operators (square brackets)*. A queue has 2 functions - *enqueue* which adds a new integer to the back of the queue, and *dequeue* which removes the integer at the front of the queue, moves everything up by one position, and returns the integer that was removed. Assume that front and back are already pointing to the front and back of the queue.

```
static int queue[100];
static int *front;
static int *back;

/* Add ‘value’ to the back of the queue. If the queue is full,
   do nothing. */
void enqueue(int value)
{







}

/* Remove the element at the front of the queue, and move
   everything else forward. Return the element that was removed */
int dequeue()
{













}
```

9. Dynamic Memory Allocation *(20 points)*
In object oriented languages like java, creating a new instance of an object involves calling its constructor method. This method dynamically allocates the memory needed for the instance, sets the initial values, and returns a pointer to it. If you can't complete one of the parts, you can still do the others - just assume that whatever you made in the part you didn't complete works when writing your solution to the others.

Part a (4 points)
Define a struct for a person. A person has a name (string), an age (int), a mother, and a father. The name string should be dynamically allocated as a char array, and the mother and father are pointers to other person structures.

Part b (6 points)
Write the constructor function `new_person`. This function should have 3 parameters (the name, the length of the name, and the age). It should allocate a new person from the heap, dynamically allocate a char array for the name, initialize all of its variables, copy the string (no buffer overflows!), and return a pointer to the newly created struct. If an error occurs, return NULL.

Part c (4 points)
Write a snippet of code that creates three persons, Alice, Bob, and Carol, and sets Carol's mother to Alice and Carol's father to Bob. (You do not have to check for errors).

Part d (6 points)
Write a destructor function `delete_person`. This function should have 1 parameter, a pointer to a dynamically allocated person structure. This function should free all of the memory allocated during the constructor function.