CMPU-377 – Parallel Programming
Spring 2019

Assigned: Wed, 1/23/2019
Due: Fri, 2/01/2019

Programming Assignment 1

The goals of this assignment are:

1. Acquaint yourself with programming in C by implementing a familiar algorithm
2. Learn how to use the vim editor
3. Practice compiling and executing C programs on a Linux workstation.

Useful Information:

- To compile your programs, cd (change directory) to the folder you created for this project. At the command prompt, type: gcc myprog.c
  If you get a prompt back with no other messages, an executable file named "a.out" will be created. To execute your program, at the command prompt, type: ./a.out
- You may find the scanf() function useful for reading input, the printf() function useful for printing output, and malloc() and calloc() useful for dynamic memory allocation.
- To submit your program, use the submit377 script, and specify assign1 (the assignment name) and your project folder.

Description:

Write a C program that finds the max of n numbers. Prompt the user to enter the number of numbers she wishes to search, then prompt for each number. Store the numbers in an array of integers. Your program should be broken up into multiple functions (akin to methods in Java or C++). The special function main() is where program execution begins. I've included my program header and some suggested function headers with appropriate comment blocks.

The reason that we're writing such a simple program is so that you can focus on learning (or remembering) the C Programming Language, the vim editor, and compiling and executing your programs.

You will write three (3) versions of this program.

1. The first uses arrays with array indexing (subscripts) to access the elements of the array.
2. The second uses pointers and pointer arithmetic (no [ ]'s) to access the numbers stored in the array.
3. I assume the array you used in the first two versions was statically defined to contain some MAX number of elements. The third version should dynamically allocate the array based on the user's input size specified. You must use malloc() or calloc() –it's your choice which—and be sure to free the memory you allocate before end of execution—there's no garbage collection in C.

Once you've successfully completed this assignment, you will have learned a great deal about C (arrays, pointers, input/output, and dynamic memory allocation), and you'll be ready to learn about languages that extend C to support parallel and distributed programming!

Final notes:

- This is an upper level course, and I expect you'll pick up programming in C (or refresh your knowledge of C) without too much difficulty. (Well, some bumps are to be expected—don't worry!)
- K&R is an excellent tutorial and reference, but others are useful, too.
- I'm available to help if you get stuck (i.e., don't waste too much time on any one problem!).

```
/* Program: seqmax.c
 * Author: Marc Smith
 *
 * Description:
 *   Write a program that finds the max of N numbers. The program will
 *   prompt the user for a list of numbers, which will be read from
 *   stdin using the C library function, scanf().
 *
 *   This is a first program in C.  While the problem is simple, many
 *   core features of the language will be mastered along the way.
 *
 *   This program will use arrays, which in C are just pointers.  You
 *   will write three versions of the program, one using array notation
 *   ( []'s ), a second using pointers ( * ), and a third using dynamic
 *   memory allocation via malloc() or calloc(). */

/*
 * Fill an array of int values, prompting the user from stdin;
 * echo the list of numbers entered, then find the max and print it
 */
int main()
{
      // you fill in the body!
}

/*
 * initialize n elements of numbers array to -1
 */
void init_numbers(int n, int* numbers)
{
      // you fill in the body!
}

/*
 * read numbers from stdin
 */
int read_numbers(int* numbers)
{
      // you fill in the body!
}

/*
 * print n elements of given array
 */
void print_numbers(int n, int* numbers)
{
      // you fill in the body!
}

/*
  * find max from first n numbers in given array
  */
int find_max(int n, int* numbers)
{
      // you fill in the body!
}
```