Assignment #1 - C Programming Basics
CMPSC311 - Introduction to Systems Programming
Spring 2018 - Prof. McDaniel

Due date: February 2nd, 2018 (Before 11:59pm)

50 points

In this assignment you will develop a program to manage several data types, data structures and arrays. The purpose of this exercise is to either remind or familiarize yourselves with the basics of C programming, as well as to expose yourself to the management and use of UNIX. Please read the following instructions very carefully and perform the assignment per the instructions.

[Step 1] Login to your Ubuntu virtual machine. From your virtual machine, download the starter source code provided for this assignment from Canvas website (Under Files).

[Step 2] Create a directory for your assignments and copy the file into it. Change into that directory.

```
% mkdir cmpsc311
% cd cmpsc311
% mkdir assign1
% cd assign1
% cp assign1-starter.zip assign1
% unzip assign1-starter.zip
```

- NOTE % is a convention to indicate terminal command line/prompt. Do not type.

Once unpacked, you will have the following starter files in the assign1 directory:
Makefile,
cmpsc311-sp18-assign1.c
cmpsc311-sp18-assign1-support.h
cmpsc311-sp18-assign1-support.c
problem4_data.txt
problem5_Array1.txt
problem5_Array2.txt

The Makefile [SEE READING MATERIALS ON CANVAS] contains commands to make your program from the source code. cmpsc311-sp18-assign1.c contains the main function which calls the functions you are to create as part of this exercise. The cmpsc311-sp18-assign1-support.h partially defines functions that you are to implement during this assignment (see below) in cmpsc311-sp18-assign1-support.c.

Problem #1 *(Print an Array)* [5 Points]

Write a ***recursive*** function **printArray** that takes an array and the size of the array as arguments, **prints** (to the terminal) the array, and returns nothing. The function should stop processing and return when it receives an array of size zero.

Problem #2 *(Print a String Backward)* [5 Points]

Write a ***recursive*** function **stringReverse** that takes a character array as an argument, ***prints*** it back to front (to the terminal) and returns nothing. The function should stop processing and return when the terminating null character of the string is encountered.

Problem #3 *(Find the Minimum Value in an Array)* [10 Points]

Write a ***recursive*** function **recursiveMinimum** that takes an integer array and the array size as arguments and returns the smallest element of the array. The function should stop processing and return when it receives an array of one element. ***Find and print*** the smallest array element found to the **terminal** for the grader.

Problem #4 *(Mean and Median Program)*

Method 1 [10 Points]

With the "response_method1" array, calculate mean and medium values.

The mean value is 6.6200 and the median value is 7.0. Add code to handle incorrect results for mean and medium results and let the grader know via the terminal. Add code to let the grader know your mean and medium functions returned the correct answers via the terminal.

When creating a medium function, check that the two middle elements are averaged in an array with an even number of elements.

Method 2 to calculate mean and medium values. [5 Points]

Rather than using the "response_method1" array, load the "***problem4_data.txt***" file into an array to calculate mean and medium values. Again, let the grader know you have correct or incorrect results from your mean and medium functions.

Problem #5 *(Intersection and Union of two arrays)* [15 Points]

Load the "***problem5_Array1.txt***" file into an array. *Load* the "***problem5_Array2.txt***" file into an array. Print the values of each array. Sort, and print the sorted arrays. *Find and print* the intersection (Find which elements are common in both arrays) of the two arrays, and the union (The set of all the unique elements) of the two arrays.

Assignment #1 Grading Schema

Graders will deduct points with the following issues:

- If your code does not compile, the maximum score you can receive is 50%.
- 10 points will be deducted with **ANY** warning messages compiling your code with the Makefile. Check on the lab machines.
- You must sort arrays with your own code. You **CANNOT** use sorting functions from other libraries and those defined in stdlib.h. You will receive zero if you do.
- **YOU MUST** use the methods and types defined in cmpsc311-sp18-assign1-support.h "regardless if your method is better". Keeping the methods as is, is necessary for fair grading. Zero will be assigned to each problem (i.e. Q1, Q2 etc) required by the method\function modified.
- **DO NOT MODIFY** the arrays in problems 1-3, and mean part of problem 4. (i.e., doing inputArray[i] = newValue; would be modifying the input array). 10 points will be deducted.
- All functions and methods must have appropriate return values. 2 points will be deducted per occurrence.
- 15 points will be deducted if your code is not properly and consistently formatted.
- Your code must contain ENGLISH comments, with the following properties:
  - Block comments are required for each file (c and header) and before each method.
  - 5 points will be deducted for each occurrence where you do not comment appropriately. Appropriate comments include the purpose, assumptions, and any known issues.
  - **DO NOT USE** Trailing comments, 15 points will be deducted if you do.
  - Use Single-Line comments to describe if/else/for/while conditions. 5 points will be deducted for each condition not commented.
  - A single-line comment should be preceded by a blank line.
  - **DO NOT** comment every single line of code. 15 points will be deducted if you do.
- Your Makefile must contain Single-Line Comments describing the purpose of the Makefile tasks. 10 points will be deducted, by not commenting the makefile.
- 10% (one-day penalty) if your submitted TGZ is not named correctly (Note: CANVAS automatically appends numbers after submission) or has incorrect directory structure described below.

REMEMBER to check your code using the Linux lab machines (W204). Graders will be using these machines to grade your work.

Submitting your work

- Your work must be in a folder called assign1 (highlighted in yellow below)
- The terminal command to create a tgz file needed for MOSS is

```
tar -czvf LASTNAME-PSUEMAILID-assign1.tgz assign1
```

where LASTNAME is your last name in all capital letters and PSUEMAILID is your PSU email address without the "@psu.edu".

- You must use the tar command above in the appropriate directory. It will not work in assign1 folder, but must be used one directory below assign1. For example, your folder structure is Desktop\McDaniel\assign1. Then execute the command in Desktop\McDaniel
- One way to check your folder structure, is to use 7zip on windows. When you use 7zip GUI to inspect the tgz file, the structure looks like this

LASTNAME-PSUEMAILID-assign1.tgz\LASTNAME-PSUEMAILID-assign1.tar\assign1\

- **BEWARE**, if you double click on the tgz file within Ubuntu, sometimes the tool will skip the tar folder, i.e. you only see assign1\ and this may not be correct. Hence, to check within your Ubuntu environment, use these steps:

    Before sending the tarball, test it using the following commands (in a temporary directory -- NOT the directory you used to develop the code):

```
% tar xvzf LASTNAME-PSUEMAILID-assign1.tgz
% cd assign1
% make
... (TEST THE PROGRAM)
```

- NOTE % is a convention to indicate terminal command line/prompt. Do not type.
- Remember, submitting your work on 11:59 is late. You must submit before 11:59 to not be considered late. One minute late will be penalized.

Note: Like all assignments in this class you are prohibited from copying any content from the Internet or discussing, sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class. Consulting online sources is acceptable, but under no circumstances should *anything* be copied. Failure to abide by this requirement will result dismissal from the class as described in our course syllabus.