CS 4540: Operating Systems
Assignment #1. Warm up C Programs

Introduction
Write three small C programs, each solving one of the following problems.

Problem #1
Write a simple C program using pointers.
The program has only the main function, in which there are 2 variables: (a) `small` that can hold real values in the range [0, 1], inclusive (the square brackets on both ends of the range mean that both "border" values are included in the range); and (b) `large` that can hold only integer values in the range [10^3, 10^6].
The program should do the following:
1) Declare variables named `small` and `large` with appropriate data types, and arbitrary initial values (limit the number of digits after the decimal point to 5).
2) Print the values of `small`, `large` and their sum.
3) Define two pointers: one pointing to `small` and the other to `large`.
4) Use pointers to assign new, different values for `small` and `large`, and print `small` and `large`.
5) Print the values of both pointers (i.e., the addresses of the `small` and `large` variables).
6) Ask the user to enter new values for the above variables, and show an error message if a variable receives an out-of-range value. For example, entering 3.6 for `small` or 100 for `large` should result in an error message.

Problem #2
Write a simple C program that uses a pseudo-random number generator (PRNG) and arrays.
It creates and manipulates two one-dimensional arrays named `numbers` and `letters`, of max. size 100 each. Fill out the array `numbers` with random integers between 1 and 1000, and array `letters` with random uppercase English letters. Print out the elements of *only* even indices of numbers and letters, that is `numbers[2], numbers[4], numbers[6], …, numbers [98]` and `letters[2], letters[4], letters[6], …, letters[98].`

Problem #3
Write a simple C program that uses arrays (among others, as function parameters), variable/array pointers and addresses.
The program should do the following:
1) Create three arrays: `arrAll` with 20 elements, and `arrEven` and `arrOdd` with 10 elements each.
2) Fill out array `arrAll` with random integer numbers between 1 and 10 inclusive.
3) Distribute contents of the array `arrAll` into arrays `arrEven` and `arrOdd`, in such a way that the contents of even/odd addresses from `arrAll` go to the `arrEven`/`arrOdd` arrays, respectively.
4) Define a pointer to the array `arrEven`.

5) Create function `calcEven(int* arrPtr, int length, &sum, &mean).` In this function, the sum and the average (mean) of the values of the indicated array will be calculated, and stored into the global variables `sum` and `mean`, respectively. Use this function to calculate sum and mean for `arrEven`.
6) Display on the output screen (from the main function) the values of `sum` and `mean` calculated by `calcEven` for `arrEven`.
7) Create function `calcOdd(int arr[ ], int length).` In this function, the sum and the average (mean) of the values of `arrOdd` will be calculated, and immediately displayed (by this function) on the output screen.

SLC Report Requirements

For each program write a full SOFTWARE LIFE CYCLE (SLC) report (analogous to the SLC report presented in class).

Please note that your reporting job is easier due to the following simplifications:

1) The PROBLEM SPECIFICATION section (Step 1) should include just a copy of the given *Problem #i* (*i* = 1, …, 4) as specified above.
2) The PROGRAM STRUCTURE DESIGN section (Step 2) will have a few modules to name (Substep *2.1. Modules and Their Basic Structure*), and few modules to provide pseudocode for (Substep *2.2.Pseudocode for the Modules*); probably a single pseudocode refinement level will suffice for Substep 2.2.
3) The sections for RISK ANALYSIS (Step 3), VERIFICATION (Step 4), REFINING THE PROGRAM (Step 7), PRODUCTION (Step 8) and MAINTENANCE (Step 9) can include just 1-2 sentences (analogous in the SLC report presented in class).
4) The CODING section (Step 5) will be simple as well; probably just two code refinement levels will suffice (the first will just add function headers and trailers, as shown in class).
5) Due to the simple structures of the small programs, the TESTING section (Step 6) should be rather easy.

The rationale for requiring you to repeat the reporting process three times for Assignment 1 is assuring that you get used to it right away.

Coding, Running and Submission Requirements

1) Follow *C Code Style Guide* and the proper programming style it requires, including comments, blank lines, indentations, spaces, etc.
2) All programs must be compiled and executed on Ubuntu running within the Oracle VirtualBox.
3) Remember about *Assignment Submission Instructions* (guidelines), to be followed for each program of this assignment.
4) In addition to what *Assignment Submission Instructions* require, provide a *makefile*, and, if needed a README file explaining how to compile and run your program.

Submission Checklist

A1 Report will consist of 3 reports: one for each of the 3 programs.

For each program you need to submit:

1) the SLC report (including 9 SLC steps, with as many pseudocode and code refinements as needed);
2) *makefile*, and, if needed a README file explaining how to compile and run the program;
3) the files created by the *script* command (including program output to the terminal, if any);
4) the output files (if any in addition to the output to the terminal) and

Submit your complete assignment package (all files) via Elearning as a *zip* file.

-------- Good luck! -----