CS 700: Assignment 2

Due Date: *Feb 20 (Su) 11:59 PM* **Overdue submissions:** 25% penalty per day

Section 1

1. [50%] Write a program that allows you to examine the effects of array size and initial data order by measuring the time of the program when your favorite sort operates on an array of integers. Tests 10 arrays each time with three different array sizes (n =100,000, n =1,000,000, and n =10,000,000) and three different array orderings (random order, ascending order, and inverse order). Three test results (random time, ordered time, reverse time) should be produced for each array in a **well-formatted** fashion. A table may be used for each array size where columns represent the 10 arrays and rows represent the three tests. Ultimately, tables are printed one after another separated by a line indicating the size of the arrays. The C++ function rand() may be helpful in building the randomly ordered arrays. If you don't have a favorite sort, use function sort defined in the header <algorithm>.

Section 2

1. [50%] Design and implement a white-box test for a program that computes the sine and cosine functions in a specialized manner. This program is going to be part of an embedded system running on a processor that does not support floating-point arithmetic. The program to be tested is attached. Your job is to test the functions sin and cos; you are to assume that the functions sin0to45 and sin45to90 have already been tested.

Marking Scheme:

- **Working program [50%]** A working program which satisfies all of the requirements automatically receives 50% of the total assignment mark. Each element of non-compliance will be penalized with respect to its severity.
- **Program Structure [25%]** A program which follows principles of Object-Oriented Design and structured programming rules (procedural, modular, uses parameters) to perfection automatically receives 25% of the total assignment mark. Marks are deducted depending on severity and number of occurrences of non-compliant elements.
- **Program Documentation [15%]**
    - **Internal documentation [10%]:** Documentation should be complete and in a standard format. Every non trivial part of the code should have a *clear* comment that explains it. In addition, every method or function, including the main program should have an explicative comment header. This header includes: module name, author, date of creation and purpose. A description of parameters and method output is mandatory. Marks are deducted according to the absence of these elements.

    - **External documentation [5%]:** HTML documentations generated by a program such as Doxygen should be provided.
- **Program Style [5%]** Style refers to Occam's razor principle. Code that is needlessly tricky, obscure, or difficult to read will be marked accordingly. Program text indentation is also an element of style and must be present. Significant constant, variable and structure names must be used. Marks are deducted on the basis of the frequency of these errors.
- **Version Control System [5%]** Track of changes that made to a program should be kept in a **private** Github repository using an IDE integrated version control system**.** The teaching assistant must be invited as the only collaborator to have an access to the repository. A link to the repository and the corresponding screenshots should be included in the submission.
- All files to be submitted should be placed in a single directory and zipped together into a single file for uploading to **UR Courses**.
- Your submission for each programming question includes: (a) the source code files (.CPP and .h files); (b) a Readme.txt file containing a brief explanation of each file along with a link to the Github repository of the assignment; and (c) screenshots of testing runs and the Github repository.