

**Department of Computer Science**

**COMP232/242/2321 (First semester - Fall2016/2017)**

*Project#4 Due Date: 10 January 2017 (along with Project#3)*

The final project of the Data Structure course consists of two parts: theoretical and practical.

In the theoretical part, you are asked to write a report about 5 new sorting algorithms.

Your final report should include the following:

* A cover page
* Write 1-1.5 page about each sorting algorithm:
	+ Write the algorithm and discuss it.
	+ Mention the algorithm properties: time complexity and space complexity and various situations, stability, and whether it is in place or not.
* A summary page comparing the 5 sorting algorithm based on the properties.
* A final page contains the used references reported in a scientific way. (Please refer here for more information)
* The following algorithms shouldn’t be included in your report:
	+ Insertion
	+ Selection
	+ Bubble
	+ Shell
	+ Merge
	+ Heap
	+ Radix
	+ Quick

In the second part of this project (the practical part), you will have to implement a program to compare the performance of the reported sorting algorithms in the first part of this project (i.e., part 1).

You will have to implement a program with the following specifications:

1. Sorting algorithms included:
	1. Sorting Algorithm 1 (from your report)
	2. Sorting Algorithm 2
	3. Sorting Algorithm 3
	4. Sorting Algorithm 4
	5. Sorting Algorithm 5
	6. All

*\*\*\* The function “All” will use all sorting algorithms to sort the generated arrays in the following options*

1. Your program should generate random arrays with the following number of elements to be sorted:
	1. 1,000
	2. 10,000
	3. 100,000
	4. 500,000
	5. 1,000,000
	6. 2,000,000
2. The user shall be given the following options to generate the randomly filled arrays with elements that are:
	1. Sorted
	2. Random
	3. Reversed Order

**The user should be given the following options:**

**1.** Select one (or more) sort algorithm to sort the arrays with.
**2.** Select one (or more) of the different sizes of the array. In the case of multiple size (i.e., 1,000 and 100,000), two arrays will be generated and then filled and sorted according to the selected sorting algorithms and the initial state of the elements in the array.
**3.** Select whether the generated array contains elements that are: sorted, random, or in reverse order (i.e., numbers are from largest to smallest; and
**4.** Print report which generates a report (into a file) containing the run-time of the selected sorting algorithms on the specified arrays (with their sizes and initial element strategy)

Once the user has selected these options, then your program should run the sort algorithms on the specified arrays with their specified sizes and initial state. Your program should compute the actual time of execution to sort the arrays. The data can be printed as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  nSort | 1,000 | 10,000 | 100,000 | 500,000 | 1,000,000 | 2,000,000 |
| S.A.1 | Time | … | … | … |  |  |
| S.A.2 |  |  |  |  |  |  |
| S.A.3 |  |  |  |  |  |  |
| S.A.4 |  |  |  |  |  |  |
| S.A.5 |  |  |  |  |  |  |

Finally, you will have to include the following in a printed format:

1. What is the best sort algorithm (and why?) for the following scenarios:
	1. When *n* size is small and the initial state of the elements is
		1. Sorted
		2. Random
		3. Reverse order
	2. When *n* size is large and the initial state of the elements is
		1. Sorted
		2. Random
		3. Reverse order
2. A table containing the results of running all sorting algorithms on all size options using all possible strategies

The deadline of this assignment will take place on **10 January 2017** along with the third project. Late submissions will not be accepted for any reason. Please make sure that your application is running properly on your laptop before the lecture. Project discussions will take place in the lecture room.

**Grading policy and general notes on the projects:**

1. Your application should have all functionalities working properly. Twenty marks will be graded for the functionality of the project;
2. The following notes will make up the remaining 10 marks of the grade:
	1. There has to be adequate documentation and comments in the code (i.e., functions, loops, etc.);
	2. Your code should follow the code convention (i.e., spaces, indentations, etc.); and
	3. Your application should contain a menu to allow the user to select which option (s) he would like to run.

Good luck!