Cairo University, Faculty of Computers and Information

FACULTY OF COMPUTERS AND INFROMATION, CAIRO UNIVERSITY

CS214: Data Structures
Year 2020 – 2021

Fall 2020, First Semester

Assignment 1 – Version 1.0

Course Instructors:
Dr. Mohammad El-Ramly

Revision History

Version 1.0	Dr Mohammed El-Ramly	20 Oct. 2020	Version 1.0

CS214: Data Structures
Assignment 1 (7 marks) – Version 1.0

Cairo University, Faculty of Computers and Information

Objectives

1- Reviewing the basics of math, proofs and C++, especially classes and templates.
2- Learning about algorithm complexity.

Instructions

1. **These instructions must be followed to get the full marks. يجب اتباع هذه التعليمات بكل دقة**
2. **Deadline is Wednesday 4 Nov. @ 11 am.**
3. Students will forms teams of 2 students.
4. Please submit **only work that you did yourself**. If you copy work from your friend or book or the net **you will fail the course**. تسليم حلول منقولة من أى مصدر يؤدى إلى الرسوب فى هذا المقرر
لا تغش الحل أو تنقله من أى مصدر و اسألنى فى أى شئ لا تفهمه لكن لا تنقل الحلول من النت أو من زملائك أو أى مكان

Task 0 (0 marks)

1. Review lecture slides & chap 1 and 2 from Data Structures & Algorithms in C++ by M. Weiss.

Task 1 (2 mark)

Each student will pick one of these two groups of problems and solve it in handwriting and:

1- Student must write in very clear handwriting and both students must use the same paper & style.
2- **Student s should combine their work together in ONE report and put in a file.**

Problems for Student 1 (from Data Structures by Mark Weisss, 4th Edition)

- **1.8.c, 1.11.a, 1.12.a, 2.1** (also for student 2), **2.6, 2.7.(a, b & c) parts 1, 3, 5**

Problems for Student 1 (from Data Structures by Mark Weisss, 4th Edition)

- **1.8.d, 1.11.b, 1.12.b, 2.1** (also for student 1)**, 2.12, 2.7.(a, b & c) parts 2, 4, 6**

Task 2 (2 marks)

Classes, objects, abstraction and operator overloading.

Students should divide the work as suggested below and then should integrate their code together and make sure it works properly.

Different variations of types **`int`** and **`float`** exist in C++ and other languages. They are usually limited by minimum and maximum values. We need versions of these types with unlimited bounds. Java solves this problem by providing **`BigInteger`** and **`BigDecimal`** classes. In this problem it is required to develop a new C++ type (class) that can hold unlimited decimal integer values and performs arithmetic operations on them. You will develop in C++ a class, **`BigDecimalInt`**, that supports writing statements with extremely long integer values like these:

```
BigDecimalInt num1("12345678901234567890123456789");
BigDecimalInt num2("11345678901134567890113456789");
BigDecimalInt num3 = num2 + num1;
cout << "num1 = " << num1 << endl;
cout << "num2 = " << num2 << endl;
//23691357802369135780236913578
cout << "num2 + num1 = " << num3 << endl;
```

CS214: Data Structures
Assignment 1 (7 marks) – Version 1.0

Cairo University, Faculty of Computers and Information

(1) Design the class **`BigDecimalInt`** that has the following public interface (set of operations available to use by developers using the class):

```
(1)  BigDecimalInt (string decStr); // Initialize from string
                                    // and rejects bad input
(2)  BigDecimalInt (int decInt);    // Initialize from integer
(3)  BigDecimalInt operator+ (BigDecimalInt anotherDec);
(4)  BigDecimalInt operator= (BigDecimalInt anotherDec);
(5)  Int size();
```

You will also need to overwrite the **`<<`** operator as follows:

```
(6) friend ostream& operator << (ostream& out, BigDecimalInt b)
```

Using data encapsulation, you are free to store the digits of the big decimal integer in whatever container you like. You might store them in an array, a vector, a string or whatever. These are details that are not important to the user of your class. You will need to build + and – operations that work on the representation you chose.

(2) Implement the class **`BigDecimalInt`** and write five test cases (including –ve numbers) to test it. Implement a program that runs the test cases and verifies the result.

(3) Name your file **A1_P1_YourGroup_YourID.cpp** or **.zip** (if more than one file)

(4) **Student 1 does functions 1, 3 and 5** above and **student 2 does 2, 4, 6** and they merge work.

Task 3 – Student 1 (3 marks) – Algorithm Complexity

Studying the complexity of searching algorithms

Warning: Learn from any resource but you must write the algorithms all by yourself.

Objective: (1) Learn efficient search algorithms (2) measure algorithm complexity practically.

In this problem, you will measure the average performance of linear and binary search algorithms in terms of the number of comparisons it does to find a word. Your task is as follows:

1- Implement a generic class called searcher that approximately has the following interface:

```
loadData (....)                 // Loads data from file
int binarySearch (....)  ┐      // Looks for a given item in the
int linearSearch (....)  ┘      // data & return its index or -1
int testPerformance (..)        // Tests performance of search algorithms
```

* Add any missing functions

2- Preferably, implement search algorithms to work on vectors not arrays. Each search algorithm should (1)calculate the time taken to search for a given word and (2) the number of comparisons it did.

3- We want to measure the (1) time and (2) number of comparisons in two cases for the two algorithms using **`testPerformance`** and the given English list of words:

- **Average performance.** First when the word is found. For this case, pick a random word (use random function C++ to pick an index between 0 and last index) and then search for it in the data.
- Do this 10 times and calculate the average time and average number of comparisons.
- **Worst case performance.** Second, makeup a random non-existing word and search for it and calculate the time and number of comparisons done until algorithm returns that word is not found.
- Do this 10 times and calculate the average time and average number of comparisons.

4- Repeat the previous step using a file of 10000, 20000, 30000, ..., 80000 words and draw the results on a plot using excel or any drawing tool. **Your plot will show the graphs of the average and worst case performance of both linear and binary search.**

5- Compare these graphs with the expected theoretical complexity of the algorithms. Are they identical or similar ? (Binary search is O(log *n*) and linear search is O(*n*)).

* You are given a list of English words to work on.

** Give the function **`testPerformance`** parameters to decide how many words to load, which sorting algorithm to run and the range of words from which to pick a random word.

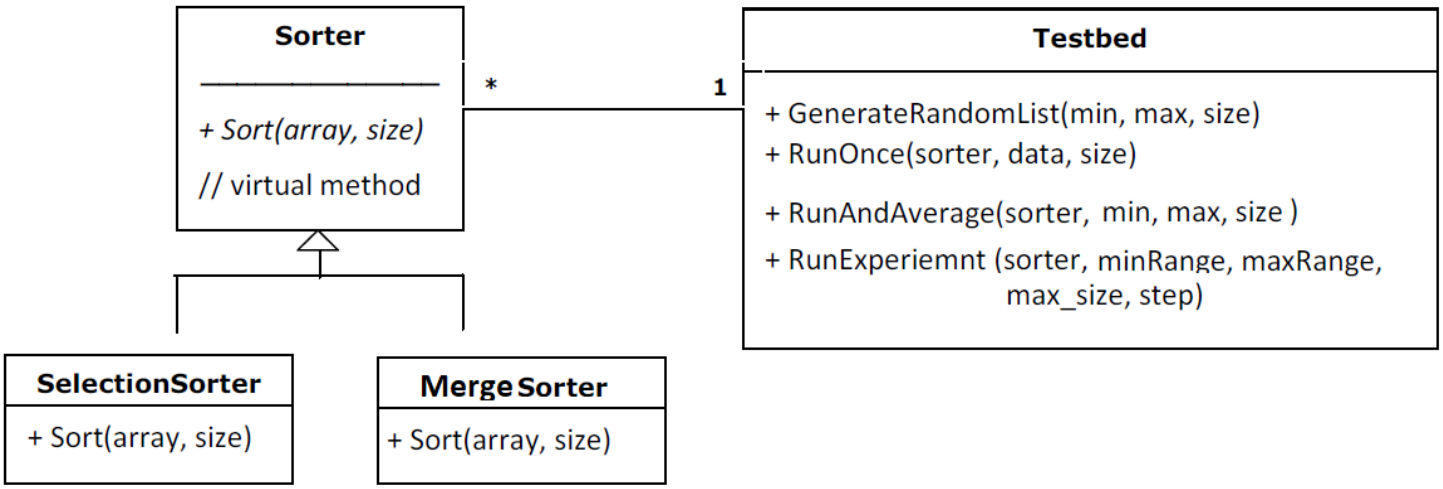
Task 3 – Student 1 (3 marks) – Algorithm Complexity

Studying the complexity of sorting algorithms

Warning: Learn from any resource but you must write the algorithms all by yourself.

Objective: (1) Learn efficient sorting algorithms (2) measure algorithm complexity practically.

1- In this problem, we will develop classes to use for testing two sorting algorithms, one O(n^2) and another O(n log n) (For example Selection sort and Merge sort) and comparing the results. The class will have methods to support experimenting and analyzing sorting algorithms performance. Below is a high level UML diagram for your classes. Add any missing details.



- **`GenerateRandomList(min, max, size)`**. Generate a given number of random integer data from a certain range. For example, one can generate a vector/array of 10000 integer numbers that fall in the range from 1 to 100000, e.g., [5554, 32300, 98000, 342, …]

CS214: Data Structures
Assignment 1 (7 marks) – Version 1.0

Cairo University, Faculty of Computers and Information

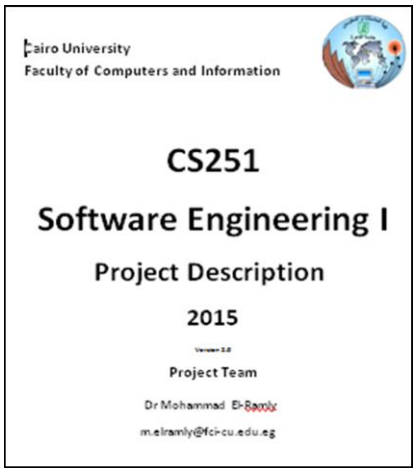
- **`RunOnce(sorter, data, size).`** Run a given sorting algorithm on a given set of data and calculate the time taken to sort the data.
- **`RunAndAverage(sorter, min, max, size).`** Run a given sorting algorithm on several sets of data of the same length and same attributes (from the same range) and calculate the average time.
- **`RunExperient (sorter, min, max, min, max_size, step).`** Develop an experiment to run a given sorting algorithm and calculate its performance on sets of different sizes (e.g., data of size 10000, 20000, etc.) as follows:
  1- All sets are generated with random values between min and max
  2- If the **`max_size`** for example is 100,000 and step is 10,000, then you will run the experiment 10 times to generate 10 data points to graph. Your data points will correspond to **`size`** or *n* = 10,000 then 20,000 then 30,000 till 100,000.
  3- For each point you will use **`RunAndAverage`** function to repeat the following 10 times and take the average. You will generate a list of random numbers of length **`size`** between **`min`** and **`max`** (say random numbers will be between 1 and 1,000,000). Then you will run the sorting algorithm on them and calculate the time. You will average the 10 times calculated and return the average. The output of the experiment goes to screen as a table with two columns; first column indicates set size, and second column indicates average time.

2- Write a **`main()`** demo to show that the function works correctly and to measure the performance of sorting algorithms using **`Testbed`** class.

3- Draw the results in a graph using Excel or any drawing tool. **Your plot will show the graphs of the average performance of the two algorithms.**

6- Compare these graphs with the expected theoretical complexity of the algorithms.

Submission Instructions


Cairo University
Faculty of Computers and Information

CS251

Software Engineering I

Project Description

2015

Project Team

Dr Mohammad El-Ramly

m.elramly@fci-cu.edu.eg

1. **Team will submit 2 things.**
2. **A handwritten report**
   - Solution of Task 1, integrated for both students.
   - The document should have a cover page like this.
   - Document should be **organized and stabled**.
3. **A zip file in acadox or other platform (TBA) containing**
   - Solution of Tasks 2, 3 integrated together for Task 2 and named: CS214-2020-A1-TaskXX.cpp
   - P**df** file (graph & performance analysis for Tasks 2 & 3)
4. Team members are expected to help each other but not do work of others.
5. **All team members must understand the details** of all solutions and be able to explain them. TA can ask any team member about any of the programs developed and its code**.**