

**Northeastern University**  
**College of Engineering**  
**Department of Electrical & Computer Engineering**

EECE7205: Fundamentals of Computer Engineering

## **Spring 2020 - Homework 4**

### **1. Instructions**

- For programming problems:
    - Your code must be well commented by explaining what the lines of your program do. Have at least one comment for every 4 lines of code.
    - You are not allowed to use any advanced C++ library unless it is clearly allowed by the problem. For example, you cannot use a library function to sort a list of data if the problem is asking you to implement an algorithm to sort the list.
    - At the beginning of your source code files write your full name, students ID, and any special compiling/running instruction (if any).
    - Test your code on the COE Linux server before submitting it:
      - a. If your program does not compile in the COE server due to incompatible text encoding format, then before uploading your source code file to the server make sure it is saved with Encoding Unicode (UTF-8). In visual studio, Save As -> Click on the arrow next to Save -> Save with Encoding -> Yes -> Unicode (UTF-8) -> Ok
      - b. Compile using `g++ -std=c++11 <filename>`
  - Submit the following to the homework assignment page on Blackboard:
    - Your homework report developed by a word processor (no handwritten or drawn contents are acceptable) and submitted as one PDF file. The report includes the following (depending on the assignment contents):
      - a. Answers to the non-programming problems.
      - b. A summary of your approach to solve the programming problems.
      - c. A summary of the skills you acquired and challenges you faced by implementing the programs.
      - d. Your recommendations of extension to the programming problems.
      - e. The screen shots of the sample run(s) of your program(s)
    - Your well-commented programs source code files (i.e., the .cc or .cpp files).
- Do NOT submit any files (e.g., the PDF report file and the source code files) as a compressed (zipped) package. Rather, upload each file individually.

**Note:** You can submit multiple attempts for this homework, however, only your last submitted attempt will be graded.

## Problem 1 (100 Points)

In this problem you will test and analyze the Insertion, Heap, and Quick sort algorithms. To carry out his task, write a C++ program with the following requirements:

1. Analyze the algorithms by sorting, in ascending order, the BST, AVG, and WST arrays. Each one of these three arrays has 1000 integers where:
  - a. BST has 1000 integers already sorted in ascending order (e.g., 10, 20, 30, ...etc.).
  - b. AVG has 1000 randomly generated integers, where each integer is between 0 and 100,000.
  - c. WST has 1000 integers sorted in descending order (e.g., 1000, 990, 980, ...etc.).

*Note:* Make sure that all algorithms are tested with identical arrays. Therefore, at the beginning of your program, create three identical versions of the BST, AVG, and WST arrays where each algorithm sorts its own version.

2. Define two global integer variables: moves and comps. Initialize these variables to zero before every call to a sorting algorithm. Add any necessary code to the algorithms' implementation to carry out the following:
  - a. Increment moves with every movement of any of the elements to be sorted from one "place" to another. However, the following is not considered a move: making a "copy" of an element that will not result eventually of that element being copied to another location in the array (i.e., the element stays in its place). Consider that the swap operation requires three movements of two elements.
  - b. Increment comps with each comparison operation on the elements to be sorted.
  - c. After each call to the sorting algorithms functions (3 algorithms  $\times$  3 arrays = 9 calls), write to a text file, sort.txt, the values of moves and comps correspond to each algorithm.
3. After calling an algorithm function to sort an array, verify that the array is sorted. One way to do that is to confirm that every element  $i$  in the array has a value less than or equal to the value in element  $i+1$  (write a function for this task and re-check your code if it returns a false verification).
4. After running your program, copy the numbers in the text file, sort.txt, to an excel sheet. Use these numbers to generate two graphs. One graph compares the number of moves needed by the three algorithms for the three cases: best, average, and worst. Another graph does the same but for the number of comparisons performed by the algorithms.

In your homework report, answer the following:

- i. Why does the Insertion sort algorithm result in zero moves when sorting an already sorted array (the best case)?
- ii. Why does the Insertion sort algorithm result in 999 comparisons when sorting an already sorted array (the best case)?
- iii. From your two excel graphs, comment on the performance of the sort algorithms under different scenarios (best, average, and worst).

## Programming Hints

- The following C++ code show you how to create a text file and write a text message in it:

```
include <iostream>
#include <fstream>
#include <stdexcept>

using namespace std;

int main() {
    ofstream outf;

    outf.open("sort.txt");
    if (outf.fail()) {
        cerr << "Error: Could not open output file\n";
        exit(1);
    }

    outf << "\t\t Hello World \n";

    outf.close(); //Close the file at the end of your program.
    return 0;
}
```

- The following C++ code show you how to generate 5 random integers. Each integer is between 0 and 100.

```
#include <chrono>

int main() {
    int A[5];

    srand(time(NULL)); //call this only once at the beginning to
                        // allow rand() to generate a different
                        // succession of random values.

    for (int i = 0; i < 5 ; i++)
        A[i] = rand() % 100;

    return 0;
}
```