COP3502C Computer Science 1
Sorting Algorithm Runtime Experiment

In this lab, you will need to write a c program and manually create an excel file based on the results from your program. You will need to submit both your code (in .c file) and the excel file data with plot in **pdf format** in the webcourses.

Experimental Study of the Execution Times of the Sorting Algorithms

In this lab, you will perform experimental studies on the sorting algorithms we have learned so far. You are asked to compare the execution times of the following sorting algorithms:

selection sort, bubble sort, insertion sort, merge sort, and quick sort

All of these algorithm source codes in c are available in your main course webcourses. So, copy them as you need.

In order to do the comparison, you will need to:

1) Create 6 arrays of different sizes. The sizes of the arrays will increase with a fixed length.
2) For each array, fill it with random numbers using random number generator.
3) Pass this array to each sorting algorithm (copy the original array before passing it to the sorting algorithm as the sorting algorithm will change your array)
4) ***Write down the execution time for each algorithm in an excel file and plot them to see how the execution time changes with the size of the data.***

A good idea could be creating an array of sizes that can be used to generate a particular number of data for the experimental array at each phase. For example, you can create an array just to store the size of each array you are going to create. Sizes array can include [10000, 20000, 30000, 40000, 50000, 100000]. This array can be useful to generate 1000 integers, 10K integers, and so on.

You may find the following hints useful:

1) To generate random numbers between [1,maxVal] you can use the rand function of <stdlib.h>:

n = rand() % maxVal + 1;

Just put this line in your for loop and store n in the array. Don't forget to include #include<stdlib.h>

2) As the execution times could be very small for smaller numbers, we would like to calculate our time in milliseonds first. To calculate the execution time you can use <time.h> library and the following technique:

```
clock_t start, end;

start = clock();
bubbleSort(sortedArray, sizes[i]);
end = clock();
long elapsed=timediff(t1,t2); //this is not a library function. See the function definition bellow
printf("\nsorting %d values takes %ld milli seconds for Bubble sort\n", sizes[i], elapsed);
```

//timediff unction should be defined by us and you can simply copy the function from here:

```
long timediff(clock_t t1, clock_t t2) {
    long elapsed;
    elapsed = ((double)t2 - t1) / CLOCKS_PER_SEC * 1000;
    return elapsed;
```

An example part of the output of your code would be like this:

```
sorting 10000 values takes 268 milli seconds for Bubble sort

sorting 10000 values takes 104 milli second for Selection sort

Sorting 10000 values takes 0 milliseconds for Merge Sort

Sorting 10000 values takes 0 milliseconds for Quick Sort

Sorting 10000 values takes 56 milliseconds for Insertion Sort

sorting 20000 values takes 1146 milli seconds for Bubble sort

sorting 20000 values takes 440 milli second for Selection sort

Sorting 20000 values takes 3 milliseconds for Merge Sort
```

An example excel file content should look like this. Submit it in pdf format:

Data Size	Run time-ABC Alorithm	Run time XYZ Algorithm
10000	2	2
20000	4	4
30000	6	8
40000	8	16
50000	10	32