CPSC 1160 SECTION 001 INSTRUCTOR: GLADYS MONAGAN

ASSIGNMENT #5: SORTING AND TIMINGS

DUE DATE WITH BRIGHTSPACE: FEBRUARY 15 AT 11:50 PM

THE PURPOSE OF THIS ASSIGNMENT IS

- to implement and study sorting algorithms
- to time code (to get empirical results)
- to determine the running time complexity of an algorithm based on empirical data

READINGS

- class notes on Brightspace as Scribbles
- §7.10
- possibly §12.3
- possibly §17.5.1

- chapter 18 on Developing Efficient Algorithms
    - §18.1 - §18.4
    - not covered: Towers of Hanoi analysis §18.4.3
    - not covered: §18.5, (§18.6, §18.7), §18.8, §18.9, §18.10
    - yes Chapter Summary

- chapter 19 on Sorting §19.1 - §19.5
    - not covered §19.6, §19.7, §19.8

IMPLEMENT

1) Implement a function that fills an array with one of the three types of data

a) random data
b) data in ascending order
c) data in descending order

Use the algorithm insertion sort provided in the file timings.cpp

```
    void insertionSort(int A[], int n);
```

Use the algorithm quick sort provided in the file timings.cpp

```
    void quickSort(int A[], int n);
```

Use the algorithm merge sort provided in the file timings.cpp

```
    void mergeSort(int A[], int W[], int n);
```

with `W` as an array of working storage (`W` has the same length as the array A)[1]:
declare W in the main function just like you should have declared A in the main function.

2) Time all 3 algorithms with arrays of data filled with a) b) and c) data

So, in total there are 9 timings per n where n is the size of the array.
Choose the values of n appropriately starting with n=1000
(suggestion, double n each time)

3) For each of the 3 sorting methods and each of the 3 data types, determine if the cost is O(n), O(n $\log_2$n) or O(n^2) and identify if this is the best case, worst case, or average case of the algorithm.

To get an accurate measure for the time for small values of `n` (and by small we mean 1000), you will need to repeat the experiment `m` times, that is, time how long it takes to sort an array of `n` values `m` times and then divide the total time `T` by `m`.
For example, for n=1000, you may need to repeat the experiment `m=100` times so that the total time is at least 1 time unit.
For the fast algorithms, you may need to use `m=1000` or larger, to get an accurate time `T`.
Your timings should NOT be 0 seconds.

[1] Do not use dynamic arrays the way it's done in the textbook.

TO SUBMIT WITH BRIGHTSPACE AS A SINGLE ZIP FILE:

A) The file timings.cpp **with your code**

The sorting routines have been implemented for you and are surrounded by

```
//===================

//===================
```

So leave that code as is and add your code before and/or after.

Leave in your source code the timing commands so that we can run your code and reproduce your timings and computations.
Produce all the data with your program, i.e. do not just produce some timings and do the rest of the calculations by hand.
Comment your source code appropriately and modularize as needed.
Remember to name your constants.
Declare the main array A to be sorted (and possibly another 'original array R of random data') and the working array W in the main function of the program as (static) arrays i.e. the types of arrays we have seen in the course so far.

Do not use *dynamic arrays* nor *variable length arrays*.

B) A pdf document

1. with the timings and results which are the output of your program: screenshots will work well if you can insert them into your document
2. for each of the 3 sorting methods using several values of n

   based on your timings and calculations of point 1.

   identify which is the best case, worst case, or average case of the algorithm.
   Determine the complexity: is it O(n), O(n $\log_2$n) or O(n^2)?
   You can do, for instance, what we did in class in Lecture 10.2 where we divided the time function by n or n^2 … to look for a constant.
   Get the computer to do the mathematical operations and include them in your results.

3. When does the best case occur for each of the sorting algorithms? Why?
   And the worst and average cases? Why?
   Discuss this.
   Explain your methodology of point 2 as to how you came up with the complexity based on your results.