Task 1 – Information Gathering

1. Search and gather existing implementations (i.e., source codes) of different sorting algorithms that are made available for free on the internet. You need to gather **at least six sorting algorithms**. You may also write your own implementation. You must include the following algorithms in your experiments: (1) **bubble sort**, (2) **insertion sort**, (3) **selection sort**, and (4) **merge sort**. It is up to you to choose the other two algorithms. For the two algorithms you choose, please provide pseudocode and a theoretical run-time analysis using the Big-oh notation.
2. Make sure that the sorting algorithms are all written in the **same programming language**.
3. Cite the source of the codes, and if available, the author/creator.
4. Study the implementation and test its correctness on a few simple examples.

Task 2 – Coding

1. The source code for each sorting algorithm must be saved on a separate file. For example, the code for merge sort function will be in the file "merge.c". This file should not contain a *main()* function.
2. Create a separate source code file that will contain a single function *GenerateData(**A**, **n**)* where ***A*** corresponds to the array and ***n*** is the array size. It should generate random ***n*** integers to be sorted.
3. Create a separate source file that will contain the *main()* function. This source file should *#include* the other files (i.e., for generating data, and sort functions). The *main()* function should do the following:

   For each value of ***n*** in ***N*** = {10,100,1000,10000,...},

   - Call *GenerateData(**A**, **n**)*
   - For each sorting algorithm,
     - For ***r*** = 1 at least ***r*** = *10* number of runs,
       - Get the start CPU time
       - Call the sort function. For example, *InsertionSort(**A**, **n**)*
       - Get the end CPU time
       - Compute the *machine execution time (**MET**) = end CPU time – start CPU time*
       - Record the ***MET***
     - Compute and record the average MET

Here is **sample C code** ↓ **(https://dlsu.instructure.com/courses/89973/files/8392697/download?download_frd=1)** for getting CPU time.

Task 3 – Testing

1. Run each sorting algorithm for different and increasing values of ***n***.
2. For each value of ***n***, execute each sorting algorithm at least ***r*** = *10* times.
3. For each sorting algorithm, record the corresponding average ***MET*** based on ***r*** runs for each value of ***n***.
4. Try to increase ***n*** to a value that can still be handled by the implementation.
5. For each ***n***, all sorting algorithms must use same input array data values – refer to Task 2 description.

For fair comparison, all tests should be made using the same machine.