# Assignment 10 - Module 10- Sorting Algorithms

[Re-submit Assignment](https://dvc.instructure.com/courses/27283/assignments/260243?module_item_id=506327)

Benchmark each of the sorting methods listed below. 

* Insertion Sort
* Bubble Sort
* Selection Sort
* Heap Sort.
* Quick Sort.
* Merge Sort.

Benchmark each of the above sorting methods for data sizes of 10000, 20000, 30000, 40000 and 50000. Display the results in a table as shown below. The table should have rows and columns. However, the rows and columns need not be separated by lines. Add two more columns for Selection and Insertion Sorts  
  

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Size | Heap Sort Time In Seconds | Merge Sort Time In Seconds | Quick Sort Time In Seconds | Bubble Sort |
| 100000 |  |  |  |  |
| 200000 |  |  |  |  |
| 300000 |  |  |  |  |
| 400000 |  |  |  |  |
| 500000 |  |  |  |  |

Notes: 

* Do not use a local array for keeping data values. Use a global array for this purpose. You can use dynamically allocated arrays if you wish.
* Generate the data using a random generator and store it in a global array for use in sorting. If you use a local array of large size, you will run out of stack space.
* Calculate the time taken by a sort routine in seconds as below:

#include <ctime>

clock\_t start, finish;   
start =clock( ); //time in milliseconds   
sort( );   
finish=clock( ); //time in milliseconds   
//the constant CLOCKS\_PER\_SEC below is equal to 1000   
double duration = (double) ( (finish-start)/CLOCKS\_PER\_SEC ); //time in secs.   
cout <<duration <<endl;