# Programming Assignment #1

**Due at 11:59pm, Tuesday, March 15**

**All the codes should be written in c or c++ for linux and commented appropriately for major steps/functions**

**Code that does not compile will not be graded and get a 0 automatically**

**The codes should be submitted as a single zipped file through Blackboard**

## Implement the Quick-Sort Algorithm using C or C++. (100 pts)

**Requirements:**

- Your code should be able to read an input ASCII file named 'input.txt' which contains the unsorted float-point numbers separated by blank space
- Your code will produce an output ASCII file named 'output.txt' contains the sorted float-point numbers separated by blank space
- Your code should output the execution time for running Quicksort excluding time of input/output.
- A script file or readme file including the instructions to compile and run the code should be submitted together with the codes

## Extra Credit: Empirical Analysis of Algorithm using C or C++. (20 pts)

**Task1**: Write a C or C++ code to study time complexity of QuickSort using different input size 10, 100, 1000, 10000, and 100000 (the number of unsorted float-point numbers). (10 pts)

**Requirements:**

- For each input size, you need to generate 100 input files randomly. You can use any random number generator to create an input file.
- For each input file, run QuickSort and record the execution time for running Quicksort excluding time of input/output.
- Compute the average running time for each input size
- A script file or readme file including the instructions to compile and run the code should be submitted together with the codes

**Task2**: Show the average running times in a plot, where x axis represents the input size and the y axis represents the time. You will have a curve for QuickSort, where a point on the curve representing the average running time for an input size. You can draw the plot manually or using any software. (10 pts)