1 Programming Assignment 1: Sorting

In this assignment we will compare the performance of the following sorting algorithms in practice. Use array-based data structures only:

1. Selection sort
2. Insertion sort
3. Quicksort (We will implement two flavors: Pivot on the last element and Randomized Pivot)

Deliverables:

1. **Data Set creation**
   To test the performance of the different sorting algorithms, we will be creating data sets of 10,000,000, 100,000,000, 500,000,000 and 1,000,000,000 numbers.

   You need to create five data set of random numbers between 0 and 1000 for each of the above input size’s. Be sure to seed your random number generator correctly! You need to submit one text file for each of the above input sizes and the source code you wrote to generate these files. **Deadline: 9/21 11:59 pm**

   **Timing:** You have been provided with an example of the timing code below. You need to also submit a brief report describing the `timeval` struct. Use man on unix or the internet to look up this struct.

2. **Insertion Sort Vs. Selection Sort (Due, Midnight, Monday September 23rd)**
   You will have 5 random samples of each input size from Deliverbale 1 above. You need to implement two functions one for Insertion Sort and one for Selection Sort, run them on all the data sets for a given size and compute the average time taken for that size of input. See the section below for information on how to time your code.

   You will submit the source code for both these algorithms (including the timing code) and a table with the time taken by each algorithm on the different sizes of input. Also include a plot of this data.

3. **Quicksort,(Due Midnight, Friday, September 27)**
   For this deliverable, you will submit the source code for both flavors of quicksort (pivot on the last element and randomized pivot). Also
submit a table with the average running time for each input size of the data set. Additionally submit a plot for all four algorithms (selection, insertion and both quicksort’s) with the input size on the X-axis and the running time on the Y-axis.

4. **Tuning (Due Midnight, Friday September 27)** Is there anything you can do to further optimize quicksort? Discuss your ideas.

1.1 **How to time a program?**

**Program Skeleton:** For this assignment, the framework that follows may be used for timing all the algorithms.

```cpp
#include the relevant headers
#include<iostream>
#include<ctime>
#include<cstdlib>
#include<sys/time.h>
using namespace std;

int main()
{
    ....
    ....
    ....

    //define the time struct
    struct timeval start, stop;

    //initialize the time struct
    start.tv_usec=0; stop.tv_usec=0;

    //call gettimeofday() before the function call
    gettimeofday(&start,NULL);
    //call bubble sort
    bubbleSort(numbers,N);
    //call gettimeofday() after the function call
    gettimeofday(&stop, NULL);
```
//calculate the difference
cout<<"Time to sort for Bubble Sort is: " <<
   ((stop.tv_sec - start.tv_sec)*1000000 +
    stop.tv_usec - start.tv_usec) <<endl;

.....
.....
}