CENG 301: Data Structures and Algorithms
Assignment 2
Fall 2013
Due date : 01.12.2013

General Rules

- You will submit you homework electronically using Ceng Homework Submission System (https://cow.ceng.metu.edu.tr).
  Login name : e1234567 (your id)
  Password : e1234567 (your id)
- Check your account and **change your password as soon as possible**. Otherwise your homework can be stolen, modified etc. This is your responsibility.Your homework must be named as "**Ceng301hw2_yourid.c**"
- Due date is **01.12.2013, 23.55** and not subject to change.
- Late submission is allowed up to **05.12.2013** with a 10% deduction penalty per day.
- **Cheating Policy** : No teaming up is allowed. The homework has to be done individually. All students in this class are bounded by the Department of Computer Engineering Guidelines for Students of Honor Code of the Faculty of the Engineering, and we have zero tolerance to cheating.
- **"Before announcing your homework grades, we will invite some randomly chosen students to explain their codes. If invited students do not come to explain their work, or could not convince us that their submission is their own work they will get zero."**
- Write your name and ID at the very top of your code.
- Programs must be written in **C language** and be ANSI C compliant. Your codes will be tested on DevC++. If you are a UNIX user you can test your codes using gcc, otherwise use DevC++.
- If you encounter any problems while submitting your homework or any questions about homework, send an e-mail to myoldas@ceng.metu.edu.tr.

Assignment

In this homework, you are required to compare three sorting algorithms given below:

1. Bubble Sort
2. Insertion Sort
3. Non-recursive Quicksort

Specifications

Write a C program that generates a profile of execution times of the sorting algorithms above. You can use clock() function from <time.h> header file to get CPU time. You call the clock() function at the beginning and end of the interval you want to time, subtract the values, and then divide by CLOCKS_PER_SEC (the number of clock ticks per second) to get processor time.

1. You will expected to write a program that executes the following tasks,

2. Create an array of randomly created 100000 doubles numbers. Each time you will sort (n/10)th (n=1,2,3,...10) of the numbers you generated, using the sorting algorithms and measure the execution time of each sorting method and keep these results in a matrix as shown in Table-1. For example, use the first 10000 elements from the original array in the first pass. Sort the first 20000 elements from the original array in the second pass and sort the first 30000 elements of the original array in the third pass etc. to produce table shown below.

Store the execution times of the first pass on the first row of a matrix, store the execution times of the second pass on the second row and the execution times of the 3rd pass on the third row of the matrix, etc.

You can increase or decrease the size of the array and/or the count of numbers in each sorting segment depending on the speed of your computer.

3. Write the matrix on the text file called "profile.txt". The numbers on each row shall be separated by at least a blank.

Note that you should sort the data in **increasing order!**

Calculation of execution (running) time: You will measure **only the execution time of the algorithm**, excluding read and write operations of your program.

The output file "profile.txt" should be as follows:

N	Bubble Sort (ms)	Insertion Sort (ms)	Quicksort (ms)
10000	1.75	1.62	1.25
20000	…..	…..	…..
30000	…..	…..	…..
…..	…..	…..	…..
…..	…..	…..	…..
…..	…..	…..	…..
…..	…..	…..	…..
…..	…..	…..	…..
…..	…..	…..	…..
100000	…..	…..	…..

Table-1. The execution times of algorithms

When program starts, user must see a menu given as follows,

1. Create 100000 random doubles.
2. Sort the numbers with 3 algorithms and save execution time in a matrix.
3. Save the data in the matrix into a file.
4. Plot the execution times versus n.
5. Exit the program.

If user selects 2nd or 3rd choices before 1st choice, a warning will be shown as "You can't select choice 2 or choice 3 before choice 1". If user selects 3rd choice before 2nd choice, a warning will be shown as "You can't select choice 3 before choice 2". When user selects

choice except 4th one, the menu will be shown again after the execution of corresponding choice is completed. When 4th one is selected, the program will halts.

4th choice is optional. If you show the graph on the screen you will get additional 30 points. You can use any header in order to implement 4th choice. If user selects 4th choice before 1st, 2nd or 3rd choices, a warning will be shown as " You can't select choice 4 before choice 2". You should show
the graph by using matrix that is created in choice 2. An example graph of execution times can be seen in Figure-1.

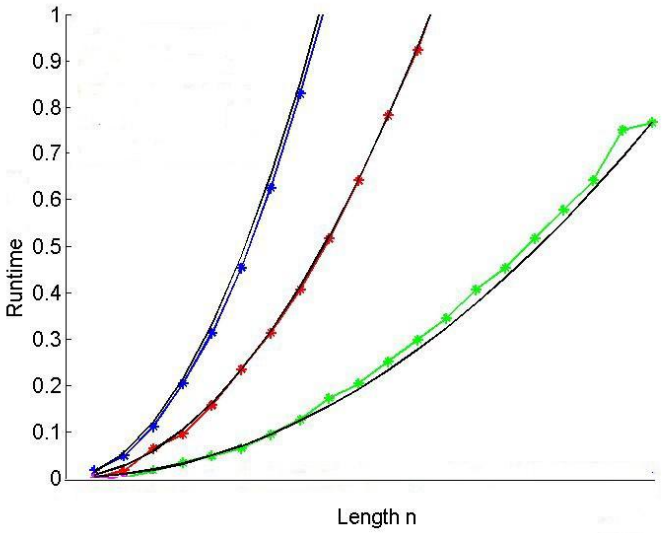

Figure-1. An Example Graph of Execution Times (The blue line represents bubble sort, the red one represents insertion sort and the green one represents quicksort).

You must not use unnecessary global variables. The functions that you have to write except main function are;

```
double createRandomNumbers() //Returns the array with random double numbers
{
        ......
}

void bublesort(double array) //Here, array is representing the array which is created in
                                function createRandomNumbers
{
        ......
}

void insertionsort(double array) //Here, array is representing the array which is
                                    created in function createRandomNumbers
{
        ......
}
```

```
void quicksort(double array) //Here, array is representing the array which is created in
                                function createRandomNumbers
{
        ......
}

double createMatrix(double array) //Here, array is representing the array which is
                                created in function createRandomNumbers. Returns the
                                matrix
{
        ......
}

void createFile(double matrix) //Here, matrix is representing the array which is
                                created in function createMatrix
{
        .......
}

void plotFigure(double matrix) //Here, matrix is representing the matrix which is
                                  created in function createMatrix
{
        .......
}

void exitProgram()
{
        ........
}
```