BLG335E, Analysis of Algorithms I, Fall 2013
Project 1
Handed out: 02.10.2013
Due: 22.10.2013, until 11.30 PM

Problem: In this project, you are expected to find closest **K** locations among **N** locations to a given location considering their coordinates on earth.

A city's location is represented with its **longitude** and **latitude**. **Longitude**s are imaginary semi-circles that intersect equator line with 90° starting from one pole to the other (North Pole and South Pole). Greenwich (in London) is accepted as a reference point (0°) for **longitude**s. There are 180 **longitude**s on the eastern side of this reference point (0° to +180°) and 180 **longitude**s on the western side (0° to -180°).

Latitudes are also imaginary circles which stand parallel to the equator starting from equator to both North and South Pole. It is known that there are 180 **latitude**s on earth and equator is accepted as a reference (0°) for them. There are 90 **latitude**s within the northern part of the equator (0° to 90°). Southern part of the equator includes the rest of the **latitude**s (0° to -90°). In the table below, some cities are shown and you may get the idea of how a city's location is represented.

City / Coordinate	Latitude	Longitude
Tokyo	35.689487	139.691706
Sydney	-33.867487	151.206990
Moscow	55.755826	37.617300
Istanbul	41.005270	28.976960
London	51.511214	-0.119824
Cape Town	-33.924869	18.424055
New York	40.714353	-74.005973
Buenos Aires	-34.603723	-58.381593
Honolulu	21.306944	-157.858333
Mexico City	19.432608	-99.133208
Stockholm	59.328930	18.064910

You are required to implement merge sort, insertion sort and linear search algorithms in order to find closest **K** locations to a given coordinate among **N** locations.

A data set (location.txt) is provided that includes city names, latitude and longitude information on each line respectively. In order to calculate the distance between two locations, you may use the formulas in the following link http://andrew.hedges.name/experiments/haversine/. You are also given a code fraction that reads the input file. You may also use this code in order not to deal with input file operations.

Your program should be run from the command line with the following format.

./studentID_AoA1_P1 N K algorithmType latitude longitude

N : Total number of locations to be sorted / searched (10, 100, 1000, 1000000)

K : Number of closest locations to be found (1, 2, 10, N/2)

algorithmType : Method to be used to solve the problem (Insertion Sort, Merge Sort or Linear Search)

latitude : Latitude information of the point that closest **K** location will be found

longitude : Longitude information of the point that closest **K** location will be found

After execution of your program, an output file should be created that lists **K** cities with their names, latitude and longitude information like the input file's format.

Detailed Instructions:

- All your code must be written in C++ using object oriented approach and able to compile and run on linux using g++.
- Submissions will be done through the Ninova server. You must submit all your program and header files. You must also submit a softcopy report.
- In your report, you are expected to analyze and compare the running times of algorithms with respect to their computational complexity.

a. Give the asymptotic upper bound on the running time for linear search, insertion sort and merge sort (which you can find in the lecture slides for sort algorithms) and show that your implementation of these algorithms fit these values.

b. Run each search methods for each different value of **N** as {10, 100, 1000, 1000000} and **K** as {1, 2, 10, N/2} calculate the average time of execution for each value of **N**.

Note: You can use the clock() function under ctime library for calculating time of execution for the search functions. Refer to http://www.cplusplus.com/reference/clibrary/ctime/clock for more details.

c. After calculating execution times you will prepare two line plots in Excel in order to visualize the runtime complexity of linear search, merge sort and insertion sort for different values of **N** and **K** (For example, you may prepare a separate plot in order to illustrate the runtime of the three algorithms for different **N** values for **K** = 1. Similarly, you may prepare 3 more plots for K = 2, 10 and N/2. This will result with 4 different plots). Then you are expected to interpret the results with respect to the asymptotic upper bounds you have given in ***a***. Indicate in which cases you would choose which algorithm. Why?

If you have any questions, please feel free to contact Res. Asst. Doğan Altan via e-mail (daltan@itu.edu.tr).