CENG 383 Project : Implementaion part. (10pts out of overall grading)

Due: 05.01.2023 (23:55)

You are expected to implement the following task using C/C++/Java. Teamwork is not allowed.

In this task you will be creating a graph from an input data files called:

1. **adjacent_cities.txt**
2. **CityDistances.txt**

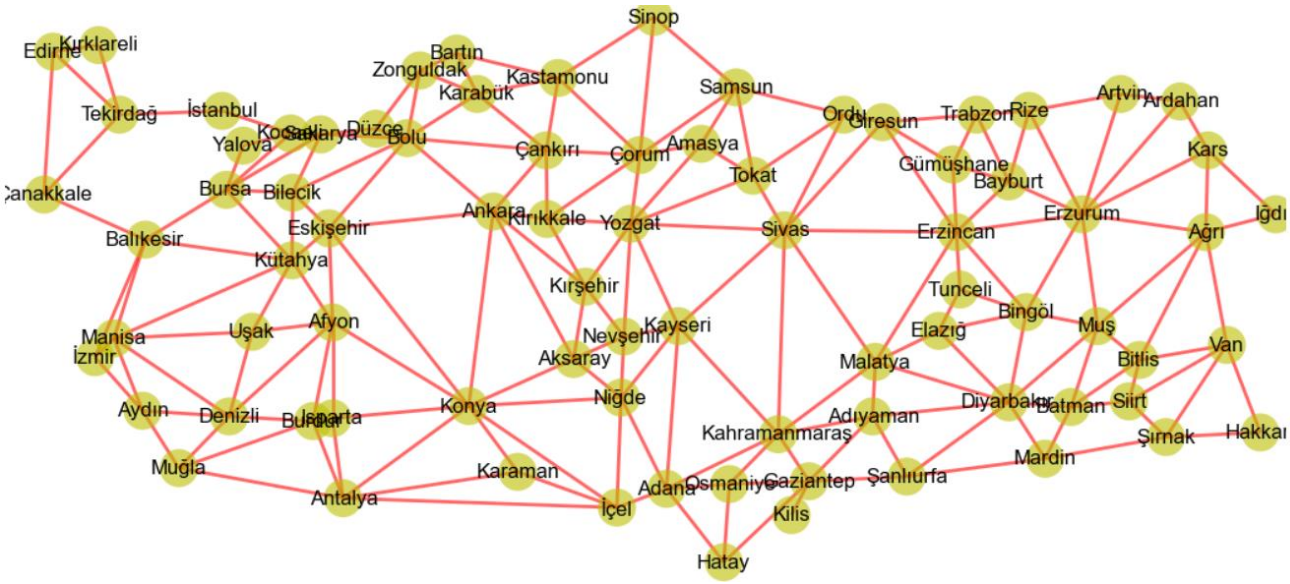
You will see a file called **adjacent_cities.txt** which contains list of neighboring cities in each line. Each line having a list of cities where the first city shares a border with the rest of the cities in the list.

Consider the following map.

First-line in " **adjacent_cities.txt** " contains

Adana,Hatay,Osmaniye,Kahramanmaras,Kayseri,Nigde,Icel

From that line, you are expected to extract neighboring relations of Adana. Next line contains the connection relation of Adiyaman, the last line has neighboring relations of Duzce. From **adjacent_cities.txt text file extraction of the city(vertex) and its connections(edges) to neighboring cities will be done. In effect, you will be creating an undirected graph that represents See the following figure.**



Then there is second text file called **CityDistances.txt.**

This text file has the distance information(weights of edges) between cities.

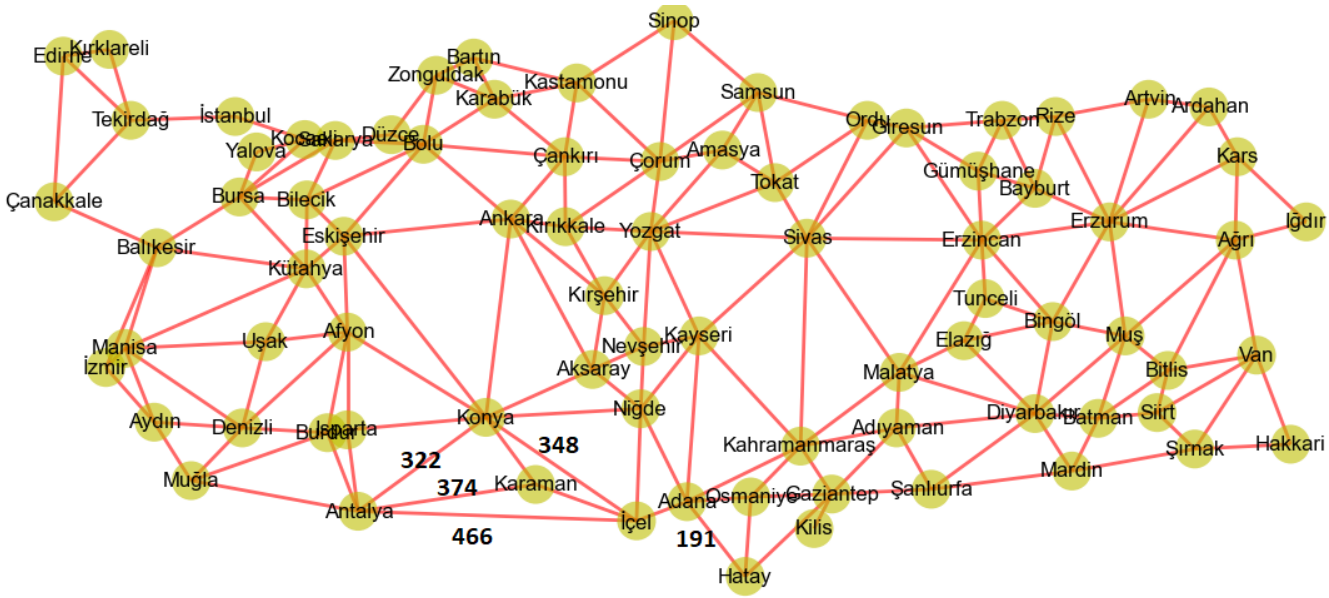
The format of the text file is as follows:

Plate Number	City Name	adana	adiyaman	afyonkarahisar	agri	amasya	ankara	…….	duzce
01	adana	0	335	575	966	603	567	…….	
02	adiyaman	335	0	910	648	632	814	…….	
03	afyonkarahisar	575	910	0	1318	597	300		
. . .	. . .	. . .	. . .	. . .		. . .	. . .	. . .	
81	duzce	735	981	375	1190	454	236	638	0

First line is the heading having plate number, city name, and list of cities from Adana to duzce.

Primarily this file is in the form of a matrix where you can find distances between cities. In this task, you will extract distances between neighboring cities that you have extracted using adjacent_cities.txt text file.

By extracting distances you can create an undirected graph that represents cities and weighted edges. (see figure below)



After creating the graph present the user with the following menu items
a. Enter city(or select)
b. Print selected(or entered) city
c. List k closest cities (to selected city)
d. Find shortest path to
x. exit

The explanation of the menu

a) Enter city: will take a plate id or name of the city and remember it as the current city
b) Show current city: will print the currently selected city.
c) Find the k closest cities from the current city, if the current has not been set, enable the user to select city, k is a number provided by the user, then your program must output k cities closest to the selected city.
d) Find the shortest path: will take a plate id or name of the city, and calculate print all shortest hops from the current city to the destination city, finding the shortest path.

You are expected to implement this problem independently (don't share your code). You can use library functions/classes such as queue, heap, stack, or list.

Grading:
50% will be given to the implementation of running code.
50% will be given to face-to-face presentation (at my office)

Graph (Structure of Class)

public:

Graph(int V) *create a* V*-vertex graph with no edges*

Graph(In in) *read a graph from input stream* in if necessary

adjacent(x, y): tests whether there is an edge from the vertex x to the vertex y;

neighbors(x): lists/vector or array of all vertices y such that there is an **edge** from the vertex x to the vertex y;

add_vertex(x): adds the vertex x, if it is not there;

remove_vertex(x): removes the vertex x, if it is there;

add_edge(x, y): adds the edge from the vertex x to the vertex y, if it is not there;

remove_edge(x, y): removes the edge from the vertex x to the vertex y, if it is there;

get_vertex_value(x): returns the value associated with the vertex x;

set_vertex_value(x, v): sets the value associated with the vertex x to v.

int V() *number of vertices*

int E() *number of edges*

String toString() *string representation*

get_edge_value(x, y): returns the value associated with the edge (x, y);

set_edge_value(x, y, v): sets the value associated with the edge (x, y) to v.

private

int V; // number of vertices

int E; // number of edges

String []labesOfNodes;

adj; // choice of adjacency lists (linked list, matrix or hash table or whatever you consider more suitable