P4- Design Document

CLASS DESIGN:

I have one class called “graph” and two structs called as “Node” and “adjacencyNode”. The vertices/nodes in my graph are stored as a linked list of nodes (struct Node used for this purpose) connected to each other. The adjacent vertices of each node are again stored as an adjacency linked list (struct adjacencyNode used for this purpose) starting from each vertice in the Node struct. The graph class contains a pointer to the graph head of struct Node, the struct Node contains a pointer to the head of the adjacency list. The shortest distance between two cities/nodes is found by using Dijkstra’s shortest path algorithm.

MEMBER VARIABLES:

Graph Class: member variables include pointer to graphHead, nodesNo(number of nodes), edgeNo(no of edges) and length.

Struct Node: member variables include pointer to adjacencyHead, next pointer of node, name(name of city node), parentNode(used for print_path function), degree(number of adjacent nodes to a particular node), shortestDistNode(shortest distance of a node), minExtractedNode(if the node is already selected – turns black),

Struct adjacencyNode: member variables include adjacencyName(name of city in the adjacency list), distance(distance between two cities), parent(used for print_path function), shortestDist(shortest distance updated in adjacency list), minExtracted(if node is already visited and selected- turns black), next pointer (of adjacencyNode).

MEMBER FUNCTIONS:

```
int insertNode(string city);
```

- Function to insert a node/city into the graph.
- Function return type is int and the parameter type is string where city is the node we insert.

```
int setd(string city1, string city2, double d);
int adjacencysetd(string city, double d);
```

- Function to set the edge distance between two cities in the graph.
- Function return type is int and the parameter type is string(cities) and double(distance).

```
string search(string city);
```

- Function to find a city in the graph with the specified city name.
- Function return type is string (city name) and the parameter type is string(specified city).

```
double findDistance(string city1, string city2);
double adjacencyDistance(string city);
```

- Function to find/print the distance set between two cities in the graph.
- Function return type is double (distance between two cities) and the parameter type is string (cities specified).

```
int findDegree(string city);
```

- Function to find the degree or number of adjacent nodes of the specified city in the graph.
- Function return type is int (distance between two cities) and the parameter type is string (city specified).

```
int graph_nodes();
```

- Function to find the number of nodes or cities in the graph.
- Function return type is int(number of nodes) and there is no parameter type.

```
int graph_edges();
```

- Function to find the number of edges or roads connecting the cities in the graph.
- Function return type is int (number of edges) and there is no parameter type.

```
void clear();
void clearNode();
```

- Function to delete all the nodes/cities in the graph.
- Function return type is void and there is no parameter type.

```
double extractMinimum();
```

- Function to find the minimum shortest distance of the nodes in the linked list. Function return type is double and there is no parameter type.

```
void relaxNode(string Vertex_name, double u);
void Update_Verticies(string city_name, double shortest_distance, string par
ent_name);
```

- 1st Function to use the relaxation theorem and update the shortest distances of cities for using Dijkstra’s algorithm. 2nd function updates the shortest distance of cities that were relaxed in 1st function. Function return type is void and the parameters are string and double

```
double findShortDist(string city1,string city2);
```

- Function to find the shortest distance between two specified cities using Dijkstra’s algorithm.
- Function return type is double (shortest distance) and the parameter types are string(cities).

```
int printPath(string city1, string city2);
void findParent(string *pathArray,string city1, string city2);
```

- 1st Function prints the shortest path between two cities. Function return type is int and parameter type is string.
- 2nd function is used to traverse from the destination node to the source/parent node. Function return type is void and the parameters types are array and string.

CONSTRUCTORS:

I have initialized three constructors which are: graph() for graph class, Node() for struct Node, adjacencyNode() for struct adjacencyNode.

DESTRUCTORS:

I have not initialized any destructors in my program.

TIME COMPLEXITY OF DIJKSTRA'S ALGORITHM:

The time complexity for my implementation of the Dijkstra's algorithm is approximately equal to O(VE). It is because my program executes Extract-Minimum function in O(V) runtime, relaxation of node in O(V) runtime. I continuously iterate through my total number of nodes to extract the minimum, and I apply relaxation theorem to all the nodes in the adjacency list. Hence, my total running time for executing Dijkstra's algorithm = $(|V| * T_{ExtractMin}) + (|E| * T_{RelaxNode}) = (|V| * |V|) + (|E| * |v|) = V^2 + VE =$ O(VE).