SCS214: Data Structures

Assignment-4

Instructions

1- Students will form teams of **2** students (**from the same lab group).**
2- Deadline of submission is **Monday May 30th at 11:55 pm.**
3- Submission will be on Blackboard.
4- No late submission is allowed.
5- No submission through e-mails.
6- **Please follow the Submission Notes found below**
7- **In case of Cheating you will get a negative grade whether you give the code to someone, take the code from someone/internet, or even send it to someone for any reason.**
8- You have to write clean code and follow a good coding style including choosing meaningful variable names.

Objective

In this assignment you will implement the dijkstra's shortest path algorithm on an undirected weighted graph. Use a min-heap to store the node labels and their current costs.

Tasks:

1- Create a struct to store the node label and its cost:

```
struct Node
{
	char label;
	int cost;
};
```

SCS214: Data Structures

Assignment-4

2- Implement a class template **MinHeap** that has the following declaration:

```
class MinHeap
{
    Node* heap; //an array of nodes
    int _size; //size of array
public:
    Node extractMin(); //returns & removes the node with minimum cost
    void buildMinHeap(Node[],int);// allocates array then builds a min-heap from an
array of struct Node with the given size
    void minHeapify(int i, int n);//restores the min-heap property for the "heap"
array using the given index and size n
    void decreaseKey(char label,int newCost);//decreases the node that has the given
label to newCost
    int parent(int i);//returns the index of the parent of i
    int getSize();//returns size of the heap
    bool inHeap(char);//checks if the node with the given label is in the heap
};
```

3- Create a class **WeightedGraph**, which stores a graph using an adjacency matrix with the following declaration:

```
class WeightedGraph
{
    int** g;
    int nVertices;
public:
    int getNVertices();//returns the number of vertices
    int getWeight(char,char);//returns weight of the edge connecting the given
vertices
    int* returnNeighbors(int v);// returns the indices of the neighbors of the vertex
v as an int array
    int numNeighbors(int v);//returns the number of neighbors of the vertex v
    void loadGraphFromFile(ifstream&);//allocates the adjacency matrix & initializes
edge weights from the specified file
    void dijkstra(char startVertex, char* prev, Node distances[] );//find the shortest
path from the start vertex to all other vertices, by filling the prev array and the
distances array
};
```

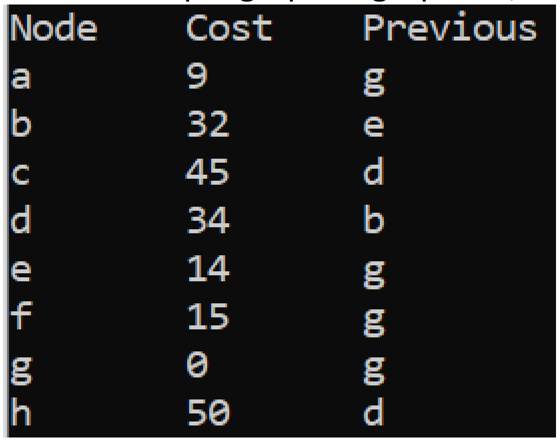
4- Your main function might look like this:

```
int main()
{
    WeightedGraph wg;
    ifstream ifile("graph.txt");
    wg.loadGraphFromFile(ifile);
    char* p;
    p = new char[wg.getNVertices()];
    Node* n;
    n=new Node[wg.getNVertices()];
    wg.dijkstra('g',p,n);
    cout<<endl<<"Node\tCost\tPrevious";
    for(int i=0;i<wg.getNVertices();i++)
    {
        cout<<endl<<n[i].label<<"\t"<<n[i].cost<<"\t"<<p[i];
    }
    ifile.close();
    return 0;
}
```

5- The file **graph.txt** will have the following format, where the first line represents number of vertices and the second line represents the number of edges. For each of the following lines, each line has the label of the first vertex, then the label of the second vertex, then the edge weight. Note that the sample graph in this file is a directed graph.

```
8
15
g a 9
g e 14
g f 15
a b 24
b d 2
b h 19
c b 6
c h 6
d c 11
d h 16
e b 18
e d 30
e f 5
f d 20
f h 44
```

6- For the sample graph in graph.txt, the output should be:



```
Node    Cost    Previous
a       9       g
b       32      e
c       45      d
d       34      b
e       14      g
f       15      g
g       0       g
h       50      d
```

SCS214: Data Structures

Assignment-4

Note that you can add any member or stand-alone functions you might need.

Submission Notes:

-Submission will be in the form of a zip file.

-Separate your implementation such that:

a. Class MinHeap declaration and implementation will be in file MinHeap.h
b. Class WeightedGraph declaration and implementation will be in file WeightedGraph.h
c. Your main function will be in main.cpp

1- Create a folder with 4 files; MinHeap.h, WeightedGraph.h, main.cpp, project.cbp/project.sln (do not include any other files)
2- Name the folder by your lab group, id1 and id2. Ex.: S5_22222222_33333333
3- Zip your folder (S5_22222222_33333333.zip)

Grading Info:

MinHeap::extractMin	**5**
MinHeap::buildMinHeap	**7**
MinHeap::minHeapify	**10**
MinHeap::decreaseKey	**10**
MinHeap::parent	**2**
MinHeap::getSize	**2**
MinHeap::inHeap	**4**
WeightedGraph::getNVertices	**2**
WeightedGraph::getWeight	**5**
WeightedGraph::returnNeighbors	**7**
WeightedGraph::numNeighbors	**5**
WeightedGraph::loadGraphFromFile	**10**
WeightedGraph::dijkstra	**21**
Total	**90**