Dijkstralab.h :

```
void dijkstra_shortest(char names[],int adj[10][10],char start,int no_of_vertices,int
max){
    int i,l,j;
    int s,e;
    for(i=0;i<no_of_vertices;i++){
        if(names[i]==start)
            s=i;
    }
    //Initial Table:
    int tab[10][3];
    for(i=0;i<no_of_vertices;i++){
        tab[i][0]=0;
        if(i!=s){
        tab[i][1]=max*2;
        tab[i][2]=-1;}
        else{
            tab[i][1]=0;
            tab[i][2]=-1;
        }
    }
    int place=s,flag,min,dist,count=0,prev_dist=0,ind;
    do{
        tab[place][0]=1;count++;
        for(j=0;j<no_of_vertices;j++){
            if(adj[place][j]!=0){
                dist=adj[place][j]+prev_dist;
                if(dist<tab[j][1]){
                    tab[j][1]=dist;
                    tab[j][2]=place;
                }
                }
            }
            //find least dist. unknown node
            min=max*2;
            for(i=0;i<no_of_vertices;i++){
                if(tab[i][0]==0){
                    if(tab[i][1]<min){
                        place=i;prev_dist=tab[place][1];min=tab[i][1];
                    }
                }
            }
        }while(count<=no_of_vertices);

        for(l=0;l<no_of_vertices;l++){
        if(l!=s){
        e=l;
        char path[10];ind=0;
        place=e;dist=tab[e][1];
        path[ind]=names[place];ind++;
        while(place!=-1){
            path[ind]=names[tab[place][2]];ind++;
            place=tab[place][2];
        }
        path[ind]=names[place];
        if(tab[e][2]==-1){
            printf("\nNo path found.");

        }
        else{
```

```
    printf("\nSHORTEST PATH: ");
    for(i=ind-2;i>=0;i--){

        printf("%c->",path[i]);
    }
    printf("\nTOTAL DISTANCE= %d\n",dist);}
}


}
}
```

Main:

```
#include<stdio.h>
#include<stdlib.h>
#include"dijkstralab.h"

int main(){
    char names[10];
    int adj[10][10];
    int no_of_vertices,i,j,max=0;
    char c;
    printf("\nEnter no. of vertices:");
    scanf("%d",&no_of_vertices);
    printf("\n\nEnter Names of vertices:");
    for(i=1;i<=no_of_vertices;i++){
        printf("\nVertice %d : ",i);
        scanf(" %c",&c);
        names[i-1]=c;
    }
```

```
        char start;
        printf("\nEnter source vertex:");
        scanf(" %c",&start);;
        printf("\nEnter the adjacency vectors for the Vertices:");
        for(i=0;i<no_of_vertices;i++){
                printf("\nVertice %c : ",names[i]);
                for(j=0;j<no_of_vertices;j++){
                        scanf(" %d",&adj[i][j]);if(adj[i][j]>max) max=adj[i][j];
                }
        }
        dijkstra_shortest(names,adj,start,no_of_vertices,max);
}
```

OUTPUT:

```
C:\Users\Gokhul\Desktop>Dijk.exe

Enter no. of vertices: 6



Enter Names of vertices:
Vertice 1 : 1

Vertice 2 : 2

Vertice 3 : 3

Vertice 4 : 4

Vertice 5 : 5
Vertice 6 : 6

Enter source vertex:1

Enter the adjacency vectors for the Vertices:
Vertice 1 : 0 5 0 6 10 0

Vertice 2 : 5 0 1 0 2 7

Vertice 3 : 0 1 0 0 0 8

Vertice 4 : 6 0 0 0 3 0

Vertice 5 : 10 2 0 3 0 4

Vertice 6 : 0 7 8 0 4 0

SHORTEST PATH: 1->2->
TOTAL DISTANCE= 5

SHORTEST PATH: 1->2->3->
TOTAL DISTANCE= 6

SHORTEST PATH: 1->4->
TOTAL DISTANCE= 6

SHORTEST PATH: 1->2->5->
TOTAL DISTANCE= 7

SHORTEST PATH: 1->2->5->6->
```

TOTAL DISTANCE= 11