```
//S Harshini-185001058
#include<stdio.h>
#include<stdlib.h>
#include"dijk.h"

int main(){
	char names[10];
	int adj[10][10];
	int no,i,j,max=0;
	char c;
	printf("\nEnter no of vertices:");
	scanf("%d",&no);
	printf("\n\nEnter names of vertices:");
	for(i=1;i<=no;i++){
		printf("\n %d : ",i);
		scanf(" %c",&c);
		names[i-1]=c;
	}
	char start;
	printf("\nEnter source vertex:");
	scanf(" %c",&start);;
	printf("\nEnter the adjacency vectors:");
	for(i=0;i<no;i++)
	{
		printf("\nVertice %c : ",names[i]);
		for(j=0;j<no;j++)
		{
			scanf(" %d",&adj[i][j]);
			if(adj[i][j]>max)
				max=adj[i][j];
		}
	}
	dijkstra(names,adj,start,no,max);
}

/* SAMPLE INPUT/OUTPUT

Enter no of vertices:6


Enter names of vertices:
 1 : a
```

2 : b

3 : c

4 : d

5 : e

6 : f

Enter source vertex:a

Enter the adjacency vectors:
Vertice a : 0 5 0 6 10 0

Vertice b : 5 0 1 0 2 7

Vertice c : 0 1 0 0 0 8

Vertice d : 6 0 0 0 3 0

Vertice e : 10 2 0 3 0 4

Vertice f : 0 7 8 0 4 0

shortest: a - b -
distance: 5

shortest: a - b - c -
distance: 6

shortest: a - d -
distance: 6

shortest: a - b - e -
distance: 7

shortest: a - b - e - f -
distance: 11
*/

/*CONTENTS OF DIJK.H FILE
void dijkstra(char names[],int adj[10][10],char start,int no,int max)

```
{
        int i,l,j;
        int s,e;
        for(i=0;i<no;i++){
                if(names[i]==start)
                        s=i;
        }

        int tab[10][3];
        for(i=0;i<no;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;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;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);

        for(l=0;l<no;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: ");
        for(i=ind-2;i>=0;i--){

                printf("%c - ",path[i]);
        }
        printf("\ndistance: %d\n",dist);}
}


}
}
*/
```