```
/*
Name            : Amod Dhopavkar
Roll No         : 33304
Batch       : K-11
Problem     : TSP using DP
*/

#include <iostream>
#include  <bits/stdc++.h>
#include <vector>
#include <string>
#define SIZE 100
using namespace std;

typedef struct Graph {
   int u, v, weight;
   string u1, v1;
}Graph;

typedef struct memory {
   int weight, prev;
}memory;

int main() {
   int num,i,j,k;
   Graph G[SIZE];
   string name,key;

   map<string,int>m;

   cout<<"\n--TSP USING DP---\n";
   cout<<"\nEnter number of cities:";
   cin>>num;

   int count=1;
   int arr[num][num];
   vector<vector<int>>vertex(num-1);
   memory M[num][num];

   for(i=0;i<num;i++) {
      cout<<"\nEnter City "<<i+1<<":";
      cin>>name;
      m.insert(pair<string,int>(name,i));
   }

   for(i=0;i<num;i++) {
```

```
    for(j=0;j<num;j++) {
        arr[i][j]=0;
    }
}

i=0;
cout<<"\nSOURCE AND DESTINATION DETAILS -->\n";

while(count==1) {
    cout<<"\nEnter the Source:";
    cin>>G[i].u1;
    G[i].u=m[G[i].u1];
    cout<<"\nEnter the Destination:";
    cin>>G[i].v1;
    G[i].v=m[G[i].v1];
    cout<<"\nEnter the Cost:";
    cin>>G[i].weight;
    arr[G[i].u][G[i].v]=G[i].weight;
    arr[G[i].v][G[i].u]=G[i].weight;
    cout<<"\nDo you want to continue(0/1)...";
    cin>>count;
    i++;
    if(count==0) {
        break;
    }
}

for(i=0;i<num;i++) {
    for(j=0;j<num;j++) {
        cout<<arr[i][j];
    }
    cout<<endl;
}

int len=i;
cout<<endl;
cout<<"Source"<<"\t"<<"Destination"<<"\t"<<"Cost";
cout<<endl;

for(i=0;i<len;i++) {
    cout<<G[i].u1<<"\t"<<G[i].v1<<"\t\t"<<G[i].weight;
    cout<<endl;
}

cout<<"\nEnter the source:";
cin>>key;
```

```
   for(i=0;i<num-1;i++) {
      for(j=0;j<num;j++) {
         if(j!= m[key]) {
            vertex[i].push_back(j);
         }
      }
   }

   for(i=0;i<num;i++) {
       for(j=0;j<num;j++) {
          M[i][j].weight = INT_MAX;
      }
   }

   cout<<vertex[0].size()<<endl;
   for(i=0;i<num-1;i++) {
      if(i==0) {                                          //calculate the cost to each vertex from
start and store it in the memory table
         for(j=0;j<vertex.size();j++) {
            M[i][vertex[j][j]].weight=arr[m[key]][vertex[j][j]];
            M[i][vertex[j][j]].prev=vertex[j][j];
            vertex[j].erase(vertex[j].begin()+j);                  //remove visited nodes
         }
      }

      else {
         int c=0;
         for(k=0;k<vertex.size();k++) {                        //calculate cost for the non visited
vertex
            int index=-1;
            int min=INT_MAX;
            for(j=0;j<vertex[k].size();j++) {
               if(c==m[key]) {
                  c++;
               }

               if(min>M[i-1][c].weight+arr[vertex[k][j]][M[i-1][c].prev]) {     //weights calculated for
previously visited node to the unvisited node
                  min=M[i-1][c].weight+arr[vertex[k][j]][M[i-1][c].prev];
                  index=j;
               }
            }
            M[i][c].weight=min;
            M[i][c].prev=vertex[k][index];                     //Add min distant node to the
memory table
```

```
                vertex[k].erase(vertex[k].begin()+index);                    //remove visited nodes
                c++;
            }

        }
    }

    for(i=0;i<num;i++) {
        if(i==m[key]) {
            continue;
        }
        M[num-1][i].weight=M[num-2][i].weight+arr[M[num-2][i].prev][m[key]];    //Add source
node to the end of the memory table
        M[num-1][i].prev=m[key];
    }

    int min=INT_MAX;
    int index1=-1;
    for(i=0;i<num;i++) {
        if(i==m[key]) {
            continue;
        }
        if(min>M[num-1][i].weight) {                                    //find optimal path
            min=M[num-1][i].weight;
            index1=i;
        }
    }
    cout<<"\nThe minimum cost is:"<<min;
    cout<<"\nThe path of the minimum cost is as follows:";
    cout<<m[key]<<"-->";

    for(i=0;i<num;i++) {
        if(i==num-1) {
            cout<<M[i][index1].prev;
        }
        else {
            cout<<M[i][index1].prev<<"-->";
        }
    }
    return 0;
}
```

Output-->

```
(base) amoddhopavkar@Amods-MacBook-Air TSP % g++ tsp.cpp -o tsp
(base) amoddhopavkar@Amods-MacBook-Air TSP % ./tsp

--TSP USING DP---

Enter number of cities:3

Enter City 1:Pune

Enter City 2:Nagpur

Enter City 3:Nagar

SOURCE AND DESTINATION DETAILS -->

Enter the Source:Nagpur

Enter the Destination:Pune

Enter the Cost:3

Do you want to continue(0/1)...1

Enter the Source:Pune

Enter the Destination:Nagar

Enter the Cost:2

Do you want to continue(0/1)...1

Enter the Source:Nagpur

Enter the Destination:Nagar

Enter the Cost:1

Do you want to continue(0/1)...0
032
301
210

Source  Destination     Cost
Nagpur  Pune            3
Pune    Nagar           2
Nagpur  Nagar           1

Enter the source:Nagpur
2

The minimum cost is:6
The path of the minimum cost is as follows:1-->0-->2-->1%
(base) amoddhopavkar@Amods-MacBook-Air TSP %
```