Name : Resham Landge
Roll No : 2339

**********Assignment No : 10**********

Title : A business house has several offices in different countries; they want to lease phone lines to connect them with each other and the phone company charges different rent to connect different pairs of cities. Business house want to connect all its offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

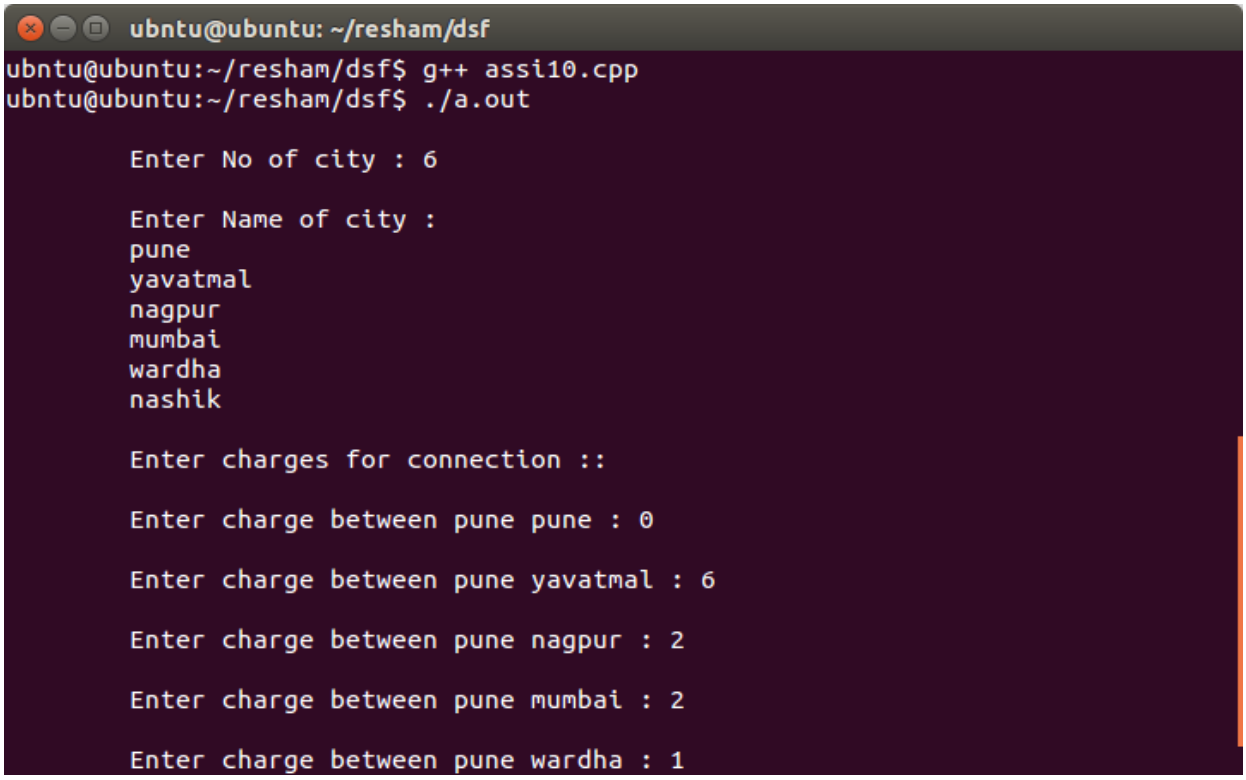
****

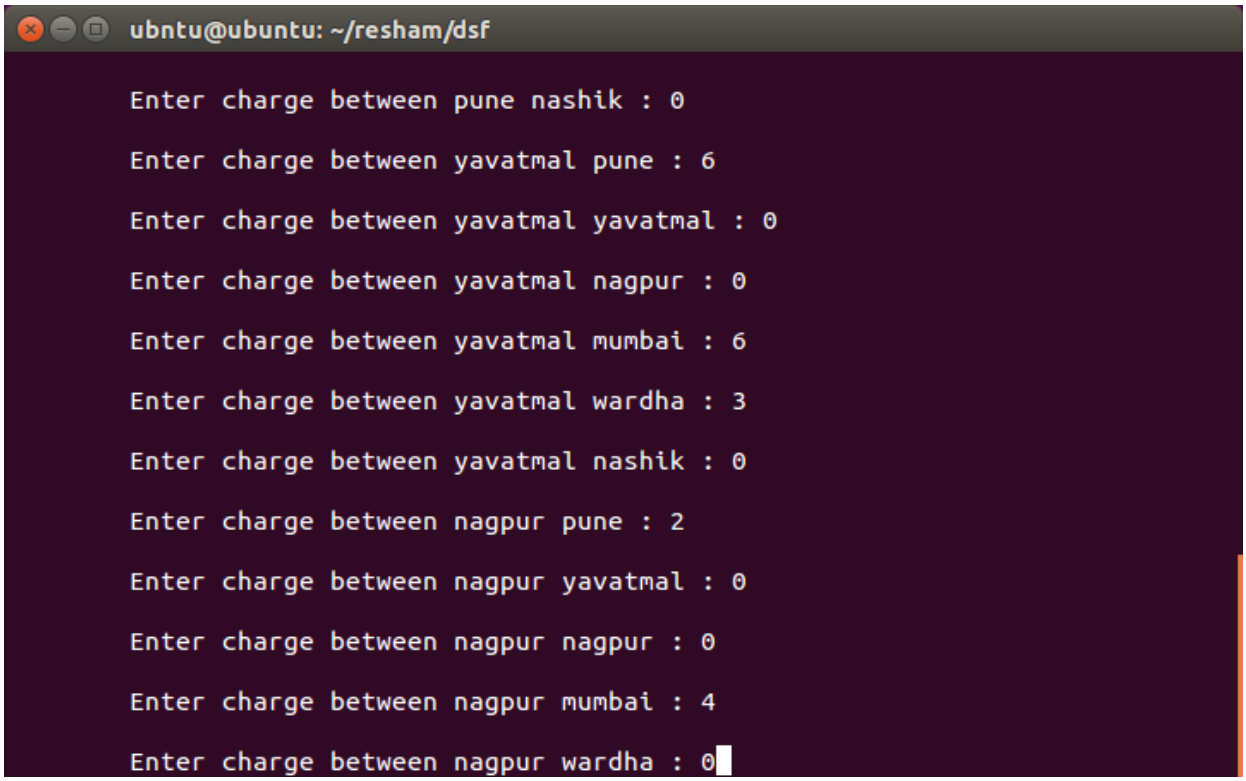
```
#include<iostream>
using namespace std;

class prims
{
        public:
        typedef struct cal
        {
                int value,from;
        }p;

        p p1[20];
        int a[20][20],h,r,i,j,k,n,min,m1,sum,vist[20],o,min1,m;
        char city[20][20];

        void create();
        void span();
        void display();
        int vis(int h);
};

void prims::create()                    // to create node into graph
{
        cout<<"\n\tEnter No of city : ";
        cin>>n;
        cout<<"\n\tEnter Name of city : ";
        for(i=0; i<n; i++)
                cin>>city[i];

        cout<<"\n\tEnter charges for connection ::\n";
        for(i=0; i<n; i++)              // for of  i for row
                for(j=0; j<n; j++)      //for of j for coloumn
                {
```

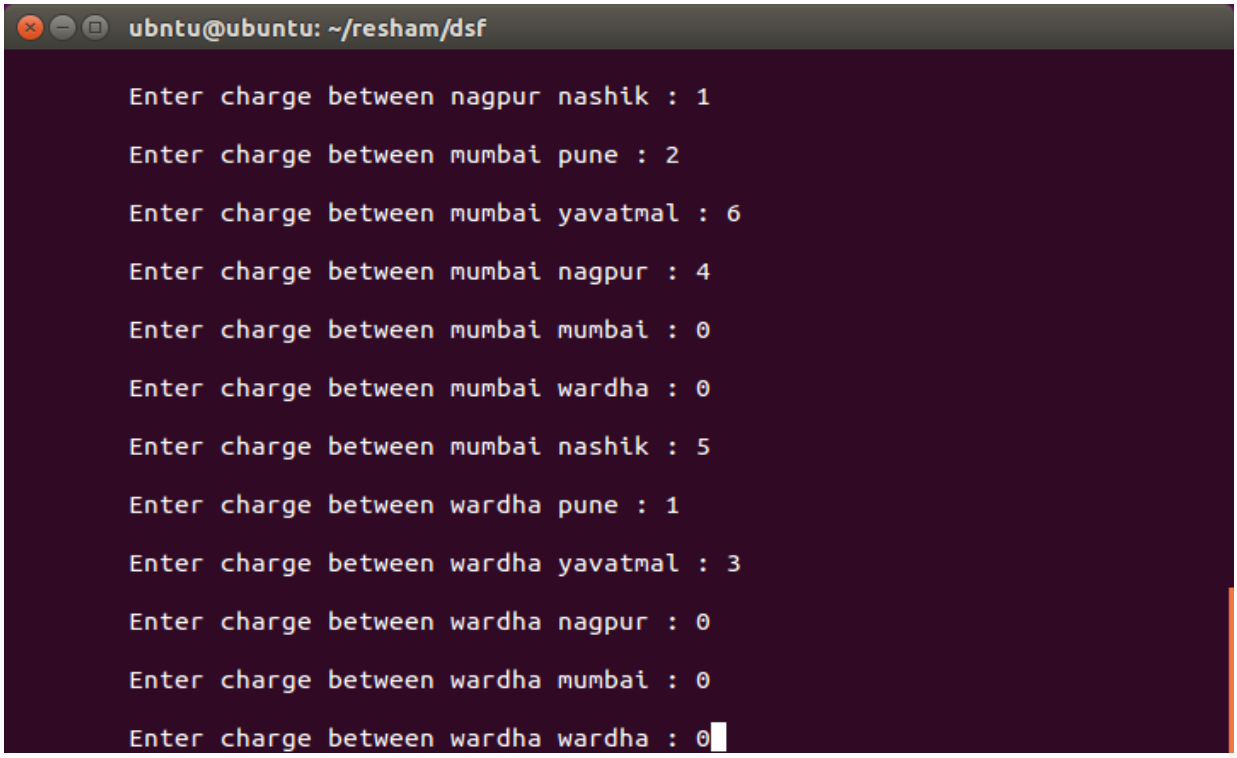
```
                cout<<"\n\tEnter charge between "<<city[i]<<" "<<city[j]<<" : ";
                cin>>a[i][j];           // to accept the input for city
            }
}

void prims::display()                   // to display the graph node
{
    cout<<"\n\tAdjacency Matrix :\n ";
    for(i=0; i<n; i++)                  // row's
    {
        for(j=0; j<n; j++)              // coloumn's
        {
            cout<<"\t"<<a[i][j]<<"\t";      //display all element
        }
        cout<<"\n";
    }
}

int prims::vis(int h)                   //function to check node is visited or not
{
    for(r=0; r<n; r++)
    {
        if(vist[r]==h)                  // to mark the visited node
            return 1;
    }
    return 0;
}

void prims::span()                      //algorithm for kruskal's
{
    for(i=0; i<n; i++)
    {
        p1[i].value=999;                // take the max value
        p1[i].from=-1;
    }

    i=0;
    m1=0;
    vist[m1++]=0;
    sum=0;

    while(m1<n)
    {
        for(j=0; j<n; j++)
```

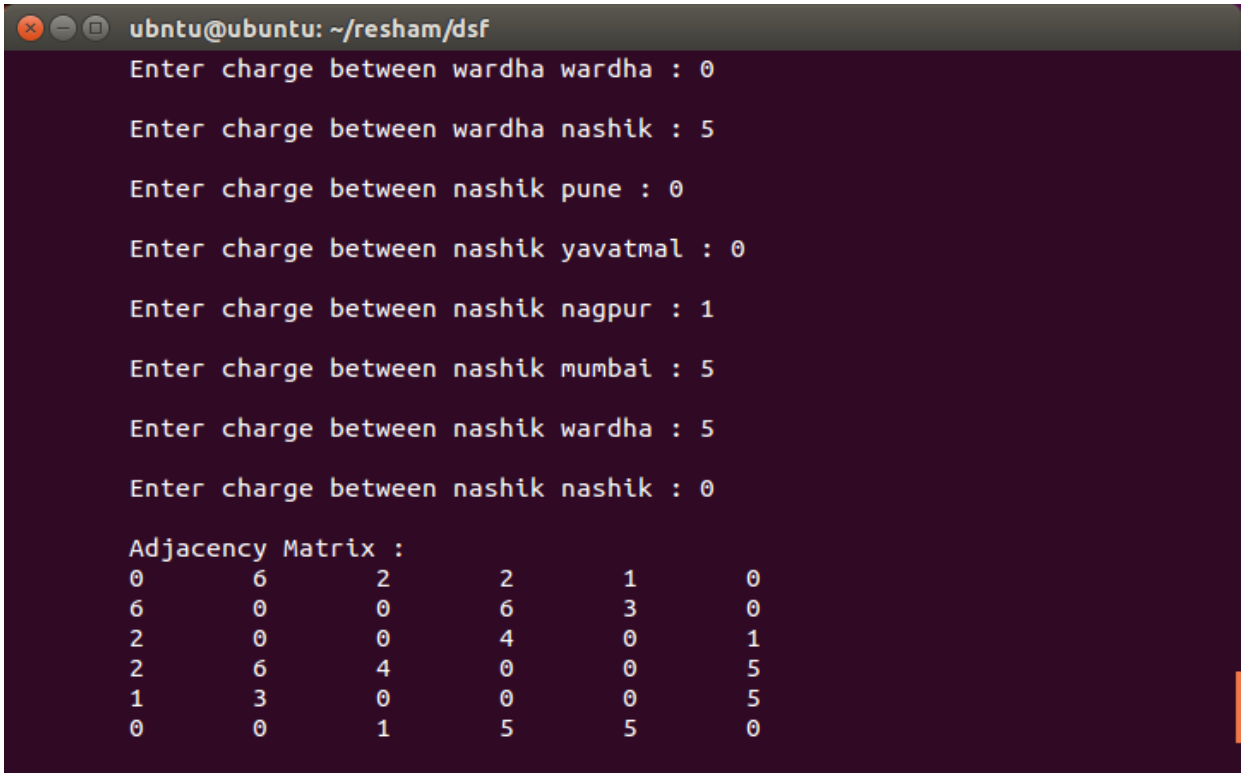
```
                {
                        if(!vis(j))                 //if node is not visited then
                        {
                                if((a[i][j]!=0)&&(p1[j].value>a[i][j])) //if the node is not zero
                                                and value of j is less than current then switch to j
                                {
                                        p1[j].value=a[i][j];
                                        p1[j].from=i;
                                }
                        }
                }

                min1=999;
                for(m=1;m<n;m++)
                {
                        if(!vis(m))                         //if node is not visited
                        {
                                if(min1>p1[m].value) //if min is less than p1[m]
                                {
                                min1=p1[m].value;     // interchange the value of m and min
                                min=m;
                                }
                        }
                }
                cout<<"\n\tEdge is visit between "<<city[min]<<"
                "<<city[p1[min].from]<<"= "<<p1[min].value;

                sum=sum+p1[min].value;      //add the min valued all node from close loop
                vist[m1++]=min;
                i=min;
        }
        cout<<"\n\tTotal cost of spanning tree = "<<sum<<"\n\n";
}

int main()
{
        prims p;
        p.create();
        p.display();
        p.span();
}
```

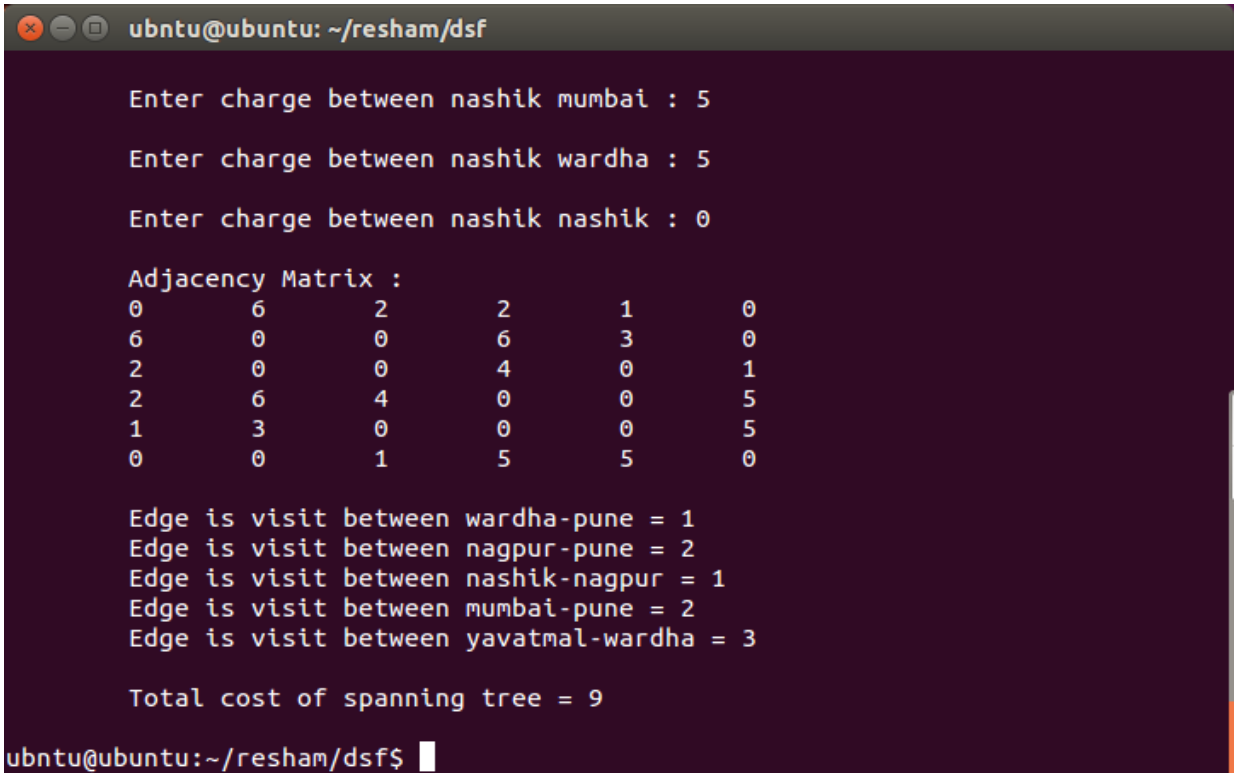
Output :


ubntu@ubuntu: ~/resham/dsf
ubntu@ubuntu:~/resham/dsf$ g++ assi10.cpp
ubntu@ubuntu:~/resham/dsf$ ./a.out
Enter No of city : 6
Enter Name of city :
pune
yavatmal
nagpur
mumbai
wardha
nashik
Enter charges for connection ::
Enter charge between pune pune : 0
Enter charge between pune yavatmal : 6
Enter charge between pune nagpur : 2
Enter charge between pune mumbai : 2
Enter charge between pune wardha : 1


ubntu@ubuntu: ~/resham/dsf
Enter charge between pune nashik : 0
Enter charge between yavatmal pune : 6
Enter charge between yavatmal yavatmal : 0
Enter charge between yavatmal nagpur : 0
Enter charge between yavatmal mumbai : 6
Enter charge between yavatmal wardha : 3
Enter charge between yavatmal nashik : 0
Enter charge between nagpur pune : 2
Enter charge between nagpur yavatmal : 0
Enter charge between nagpur nagpur : 0
Enter charge between nagpur mumbai : 4
Enter charge between nagpur wardha : 0


ubntu@ubuntu: ~/resham/dsf
Enter charge between nagpur nashik : 1
Enter charge between mumbai pune : 2
Enter charge between mumbai yavatmal : 6
Enter charge between mumbai nagpur : 4
Enter charge between mumbai mumbai : 0
Enter charge between mumbai wardha : 0
Enter charge between mumbai nashik : 5
Enter charge between wardha pune : 1
Enter charge between wardha yavatmal : 3
Enter charge between wardha nagpur : 0
Enter charge between wardha mumbai : 0
Enter charge between wardha wardha : 0


ubntu@ubuntu: ~/resham/dsf
Enter charge between wardha wardha : 0
Enter charge between wardha nashik : 5
Enter charge between nashik pune : 0
Enter charge between nashik yavatmal : 0
Enter charge between nashik nagpur : 1
Enter charge between nashik mumbai : 5
Enter charge between nashik wardha : 5
Enter charge between nashik nashik : 0
Adjacency Matrix :
0 6 2 2 1 0
6 0 0 6 3 0
2 0 0 4 0 1
2 6 4 0 0 5
1 3 0 0 0 5
0 0 1 5 5 0


ubntu@ubuntu: ~/resham/dsf
Enter charge between nashik mumbai : 5
Enter charge between nashik wardha : 5
Enter charge between nashik nashik : 0
Adjacency Matrix :
0 6 2 2 1 0
6 0 0 6 3 0
2 0 0 4 0 1
2 6 4 0 0 5
1 3 0 0 0 5
0 0 1 5 5 0
Edge is visit between wardha-pune = 1
Edge is visit between nagpur-pune = 2
Edge is visit between nashik-nagpur = 1
Edge is visit between mumbai-pune = 2
Edge is visit between yavatmal-wardha = 3
Total cost of spanning tree = 9
ubntu@ubuntu:~/resham/dsf$