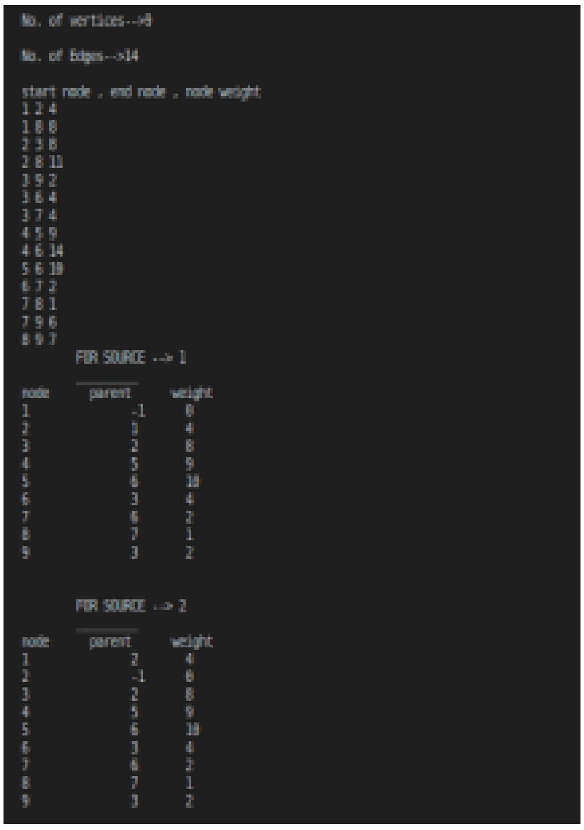
Khushi Chhatwani CSC/21/55

PRIM’S ALGORITHM :

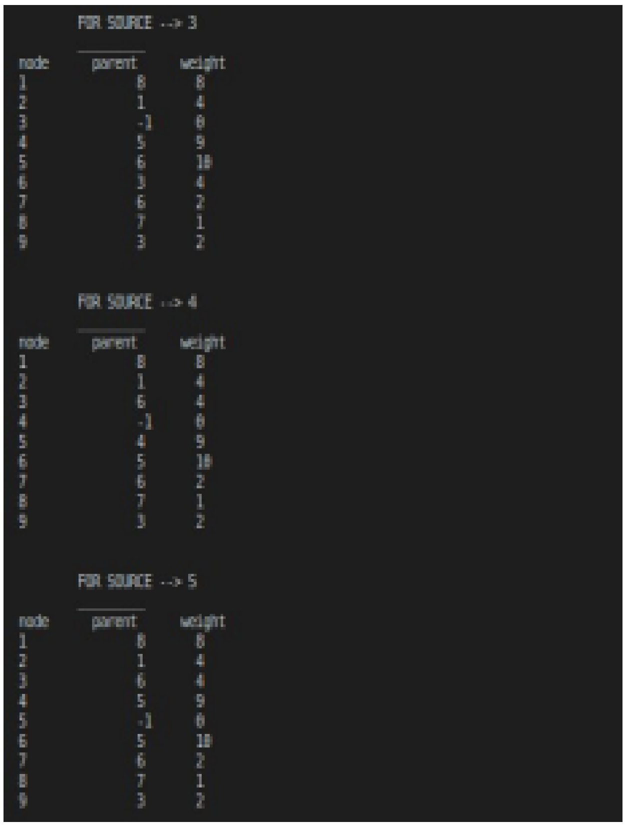
```
#include <bits/stdc++.h>
using namespace std;
class Compare
{
public:
bool operator()(pair<int, int> a, pair<int, int> b)
{
return a.second > b.second;
}
};

void primMST(unordered_map<int, list<pair<int, int>>> g, int n, int source) {
vector<int> key(n + 1, 100);
vector<bool> inMst(n + 1, 0);
vector<int> parent(n + 1, -1);
int src = source;
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
pq.push(make_pair(0, src));
key[src] = 0;

while (!pq.empty())
{
int u = pq.top().second;
pq.pop();
inMst[u] = 1;
for (auto it : g[u])
{
int v = it.first;
int w = it.second;
if (inMst[v] == 0 && w < key[v])
{
key[v] = w;
pq.push(make_pair(key[v], v));
parent[v] = u;
}
}
}
cout << "node parent weight \t \n";
for (int i = 1; i <= n; ++i)
{
cout << i << "\t\t";
cout << parent[i] << "\t";
cout << key[i] << "\t \n";
}
}
```

```cpp
int main()
{
int n;
cout<<"No. of vertices-->";
cin >> n;
int e;
cout<<"\nNo. of Edges-->";
cin >> e;
cout<<endl;
unordered_map<int, list<pair<int, int>>> g;
cout<<"start node , end node , node weight \n";
for (int i = 1; i <= e; i++)
{
int u, v, wt;
cin >> u >> v >> wt;
g[u].push_back(make_pair(v, wt));
g[v].push_back(make_pair(u, wt));
}
for(int i=1;i<=n;i++)
{
cout<<"\tFOR SOURCE --> "<<i<<endl;
cout<<"\t__________\n";
primMST(g, n, i);
cout<<"\n\n";
}
}
```

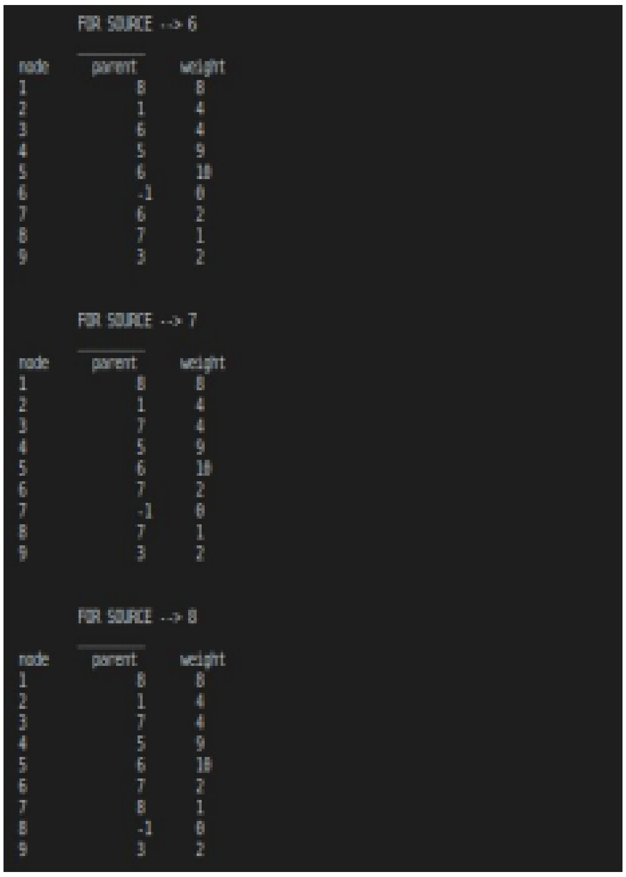
OUTPUT :



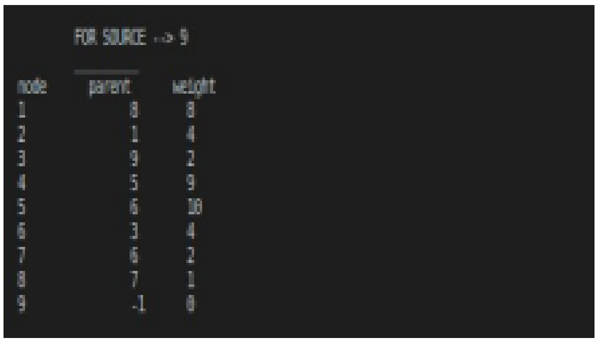
```
No. of vertices-->9

No. of Edges-->14

start node , end node , node weight
1 2 4
1 8 8
2 3 8
2 8 11
3 9 2
3 6 4
3 7 4
4 5 9
4 6 14
5 6 10
6 7 2
7 8 1
7 9 6
8 9 7
        FOR SOURCE --> 1
        __________
node        parent      weight
1               -1      0
2               1       4
3               2       8
4               5       9
5               6       10
6               3       4
7               6       2
8               7       1
9               3       2


        FOR SOURCE --> 2
        __________
node        parent      weight
1               2       4
2               -1      0
3               2       8
4               5       9
5               6       10
6               3       4
7               6       2
8               7       1
9               3       2
```



```
        FOR SOURCE --> 3

node      parent      weight
1             8        8
2             1        4
3             -1       0
4             5        9
5             6        10
6             3        4
7             6        2
8             7        1
9             3        2


        FOR SOURCE --> 4

node      parent      weight
1             8        8
2             1        4
3             6        4
4             -1       0
5             4        9
6             5        10
7             6        2
8             7        1
9             3        2


        FOR SOURCE --> 5

node      parent      weight
1             8        8
2             1        4
3             6        4
4             5        9
5             -1       0
6             5        10
7             6        2
8             7        1
9             3        2
```



```
        FOR SOURCE --> 6

node      parent      weight
1             8        8
2             1        4
3             6        4
4             5        9
5             6        10
6             -1       0
7             6        2
8             7        1
9             3        2


        FOR SOURCE --> 7

node      parent      weight
1             8        8
2             1        4
3             7        4
4             5        9
5             6        10
6             7        2
7             -1       0
8             7        1
9             3        2


        FOR SOURCE --> 8

node      parent      weight
1             8        8
2             1        4
3             7        4
4             5        9
5             6        10
6             7        2
7             8        1
8             -1       0
9             3        2
```



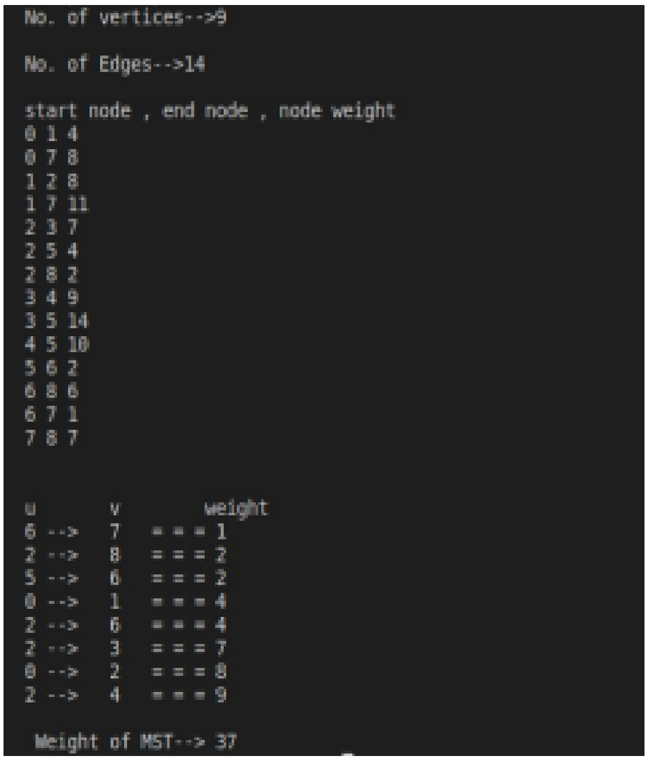
```
        FOR SOURCE --> 9

node      parent      weight
1             8        8
2             1        4
3             9        2
4             5        9
5             6        10
6             3        4
7             6        2
8             7        1
9             -1       0
```

KRUSKAL’S ALGORITHM :

```
#include<bits/stdc++.h>
using namespace std;
```

```
void makeSet(vector<int>&parent ,vector<int>&rank,int n)
{
for(int i=0;i<n;i++)
{
parent[i]=i;
rank[i]=0;
}
}
bool cmp(vector<int>a,vector<int>b)
{
return a[2]<b[2];
}
int findParent(vector<int>&parent,int i)
{
if(parent[i]==i) return i;
return parent[i]=findParent(parent,parent[i]);
}
void unionSet(int u,int v,vector<int>&parent,vector<int>&rank)


{
u=findParent(parent,u);
v=findParent(parent,v);

if(rank[u] < rank[v])
{
parent[u]=v;
}
else if(rank[v]<rank[u])
{
parent[v]=u;
}
else
{
parent[v]=u;
rank[u]++;
}
}
int MST(vector< vector<int> >&edge,int n)
{
sort(edge.begin(),edge.end(),cmp);
vector <int> parent(n);
vector<int> rank(n);
makeSet(parent,rank,n);
int weight=0;
cout<<"\n\nu\tv \t weight\n";
for(int i=0;i<edge.size();i++)
{
int u=findParent(parent,edge[i][0]);
int v=findParent(parent,edge[i][1]);
int w=edge[i][2];

if(u!=v)
{
weight+=w;
unionSet(u,v,parent,rank);
cout<<u<< " --> "<<v<< " = = = "<<w<<endl;
}
}
return weight;
}
int main()
{
int n;
cout<<"No. of vertices-->";
cin >> n;
int e;
cout<<"\nNo. of Edges-->";


cin >> e;
cout<<endl;
cout<<"start node , end node , node weight \n";
vector< vector<int> >edge;
for(int i=0;i<e;i++)
{
int u,v,w;
cin>>u>>v>>w;
vector<int>temp;
temp.push_back(u);
temp.push_back(v);
temp.push_back(w);
edge.push_back(temp);
}
int k=MST(edge,n);
cout<<"\t \n Weight of MST--> "<<k<<"\n";
}
```

OUTPUT :



```
No. of vertices-->9

No. of Edges-->14

start node , end node , node weight
0 1 4
0 7 8
1 2 8
1 7 11
2 3 7
2 5 4
2 8 2
3 4 9
3 5 14
4 5 10
5 6 2
6 8 6
6 7 1
7 8 7


u       v        weight
6 -->   7   = = = 1
2 -->   8   = = = 2
5 -->   6   = = = 2
0 -->   1   = = = 4
2 -->   6   = = = 4
2 -->   3   = = = 7
0 -->   2   = = = 8
2 -->   4   = = = 9

 Weight of MST--> 37
```