Name- Chiradeep Banik.

Enrol. No. - 20UCS176

Branch - Computer Sc. and Engineering

Section - A

Reg. No. - 2012819

Q.1

```
# include <iostream>
# include <cstring>
using namespace std;
# define  INF 9999999
# define  V 5
int  G[V][V] = {
        {0, 9, 75, 0, 0},
        {9, 0, 95, 19, 42},
        {75, 95, 0, 51, 66},
        {0, 19, 51, 0, 31},
        {0, 42, 66, 31, 0}
};

int main() {
    int no_edge = 0;
    int selected[V];
    memset ( selected, false, sizeof(selected));
    selected [0] = true
    int x, y;
    cout << "Edge : weight" << endl;
```

→

```
while (no_edge < V-1){
    int min = INF;
    x=0, y=0;
    for (int i=0; i<V; i++) {
        if(selected[i]){
            for (int j=0; j<V; j++){
                if (!selected[j] && G[i][j]){
                    if (min > G[i][j]){
                        min = G[i][j];
                        x=i;
                        y=j;
                    }
                }
            }
        }
    }
    cout << x << " - " << y << " : " << G[x][y] << endl;
    selected[y]=true;
    no_edge ++;
}
return 0;
}
```

Output

↳
```
Edge : weight
0-1  :  9
1-3  :  19
3-4  :  31
3-2  :  51
```

⟹

Q.2
```
# include <bits/stdc++.h>
using namespace std;
class DSU {
     int* parent;
     int* rank;
  public:
     DSU(int n){
          parent = new int[n];
          rank = new int[n];
          for(int i=0; i<n; i++){
               parent[i]=-1;
               rank[i] = 1;
          }
     int find(int i){
          if (parent [i] == -1)
               return i;
          return parent[i]= find(parent[i]);
     }
     void unite(int x, int y) {
          int s1= find(x);
          int s2= find(y);
          if (s1 != s2){
               if (rank[s1] < rank[s2]){
                    parent[s1]= s2;
                    rank[s2] += rank[s1];
               }else{
                    parent[s2] = s1;
                    rank[s1] += rank[s2];
               }
          }
     }
};
```
→

```
class Graph {
    vector<vector<int>> edgelist;
    int V;
public:
    Graph (int V) {
        this->V=V;
    }
    void addEdge (int x, int y, int w){
        edgelist.push_back({w,x,y});
    }
    void kruskals_mst(){
        sort(edgelist.begin(), edgelist.end());

        DSU s(V);
        int ans =0;
        cout << "Edges in MST construction "<<endl;
        for(auto edge : edgelist){
            int w= edge[0];
            int x= edge[1];
            int y= edge[2];
            If (s.find (x) != s.find(y)) {
                s.unite(x, y);
                ans += w;
                cout << x << "--" << y << "==" << w <<endl;
            }
        }
        cout << "Minimum Cost Spanning Tree: " << ans <<endl;
    }
};
```

→

```
int main(){
    Graph g(4);
    g.addEdge(0,1,10);
    g.addEdge(1,3,15);
    g.addEdge(2,3,4);
    g.addEdge(2,0,6);
    g.addEdge(0,3,5);
    g.kruskals_mst();
    return 0;
}
```

Output

```
↳ Edges in MST construction
    2 --- 3 == 4
    0 -- 3 == 5
    0 -- 1 ==10
    Minimum Cost Spanning Tree : 19
```

[illegible]