//ASSIGNMENT 7 – PRIM’S AND KRUSKAL’S ALGORITHMS

1)PRIM’S ALGORITHM

Program:

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

#define V 5
#define HIGH 999999

int main()
{
   int G[V][V]={
    {0,4,0,5,2},
    {4,0,1,3,0},
    {0,1,0,8,0},
    {5,3,8,0,2},
    {2,0,0,2,0}
   };

int v_array[V];
memset(v_array,false,sizeof(v_array));
v_array[0]=true;
cout<<"Edge : weight\n";
int no_edge=0;
while(no_edge<V-1)
{
   int min=HIGH;
   int r=0;
   int c=0;
   for(int i=0;i<5;++i)
   {
      if(v_array[i])
```

```
        {
           for(int j=0;j<5;++j)
           {
              if(min>G[i][j])
              {
                 if(!v_array[j] && G[i][j])   //G[0][0]=flase
                 {
                    min=G[i][j];
                    r=i;
                    c=j;
                 }
              }
           }
        }
     }
     cout<<" "<<r<<"-"<<c<<"  :  "<<G[r][c]<<endl;
     v_array[c]=true;
     no_edge++;

  }

  return 0;
  }
```

OUTPUT:

```
Edge : weight
 0-4  :  2
 4-3  :  2
 3-1  :  3
 1-2  :  1
```

2)KRUSKAL’S ALGORITHM

Program:

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

typedef pair<int, int> iPair;
struct Graph
{
    int V, E;
    vector< pair<int, iPair> > edges;

    Graph(int V, int E)
    {
        this->V = V;
        this->E = E;
    }

    void addEdge(int u, int v, int w)
    {
        edges.push_back({w, {u, v}});
    }
    int kruskalMST();
};

struct DisjointSets
{
    int *parent, *rnk;
    int n;

    DisjointSets(int n)
    {
        this->n = n;
        parent = new int[n+1];
        rnk = new int[n+1];
        for (int i = 0; i <= n; i++)
        {
            rnk[i] = 0;
            parent[i] = i;
        }
    }

    int find(int u)
    {
        if (u != parent[u])
            parent[u] = find(parent[u]);
        return parent[u];
    }

    void merge(int x, int y)
    {
        x = find(x), y = find(y);

        if (rnk[x] > rnk[y])
            parent[y] = x;
        else // If rnk[x] <= rnk[y]
            parent[x] = y;

        if (rnk[x] == rnk[y])
            rnk[y]++;
    }
};

int Graph::kruskalMST()
{
    int mst_wt = 0;

    sort(edges.begin(), edges.end());

    DisjointSets ds(V);

    vector< pair<int, iPair> >::iterator it;
    for (it=edges.begin(); it!=edges.end(); it++)
    {
        int u = it->second.first;
        int v = it->second.second;

        int set_u = ds.find(u);
        int set_v = ds.find(v);

        if (set_u != set_v)
        {
            cout << u << " - " << v << endl;
            mst_wt += it->first;
            ds.merge(set_u, set_v);
        }
    }

    return mst_wt;
}

int main()
{

    int V = 9, E = 14;
    Graph g(V, E);

    // making above shown graph
    g.addEdge(0, 1, 4);
    g.addEdge(0, 7, 8);
    g.addEdge(1, 2, 8);
    g.addEdge(1, 7, 11);
    g.addEdge(2, 3, 7);
    g.addEdge(2, 8, 2);
    g.addEdge(2, 5, 4);
    g.addEdge(3, 4, 9);
    g.addEdge(3, 5, 14);
    g.addEdge(4, 5, 10);
    g.addEdge(5, 6, 2);
    g.addEdge(6, 7, 1);
    g.addEdge(6, 8, 6);
    g.addEdge(7, 8, 7);

    cout << "Edges of MST are \n";
    int mst_wt = g.kruskalMST();

    cout << "\nWeight of MST is " << mst_wt<<endl;

    return 0;
}
```

OUTPUT:

```
Edges of MST are
6 - 7
2 - 8
5 - 6
0 - 1
2 - 5
2 - 3
0 - 7
3 - 4

Weight of MST is 37
```