```
#include <iostream>
#include <vector>
#include <utility>
#include <algorithm>
using namespace std;
const int MAX = 1000;
int id[MAX], nodes, edges;  //array id is use for check the parent of vertex;
pair <long long, pair<int, int> > p[MAX];

   //initialise the parent array id[]
void init()
{
    for(int i = 0;i < MAX;++i)
        id[i] = i;
}

int root(int x)
{
    while(id[x] != x)  //if x is not itself parent then update its parent
    {
        id[x] = id[id[x]];
        x = id[x];
    }
    return x; //return the parent
}

//function for union
void union1(int x, int y)
{
    int p = root(x);
    int q = root(y);
    id[p] = id[q];
}

//function to find out the edges in minimum spanning tree and its cost
long long kruskal(pair<long long, pair<int, int> > p[])
{
    int x, y;
    long long cost, minimumCost = 0;
    for(int i = 0;i < edges;++i)
    {
        x = p[i].second.first;
        y = p[i].second.second;
        cost = p[i].first;
        if(root(x) != root(y))
        {
            minimumCost += cost;
    cout<<x<<" ----> "<<y<<" :"<<p[i].first<<endl;//print the edges contain in
spanning tree
            union1(x, y);
        }
    }
    return minimumCost;
}

int main()
{
    int x, y;
    long long weight, cost, minimumCost;
```

```
    init();
    cout <<"Enter Nodes and edges"<<endl;
    cin >> nodes >> edges;

    //enter the vertex and cost of edges
    for(int i = 0;i < edges;++i)
    {
        cout<<"Enter the value of X, Y and edges"<<endl;
    cin >> x >> y >> weight;
        p[i] = make_pair(weight, make_pair(x, y));
    }

    //sort the edges according to their cost
    sort(p, p + edges);
    minimumCost = kruskal(p);
    cout <<"Minimum cost is "<< minimumCost << endl;
    return 0;
}
```