```
// IMPLEMENTATION OF KRUSKALS
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
const int MAX = 1e6-1; int
root[MAX];
const int nodes = 4, edges = 5; pair <long
long, pair<int, int> > p[MAX];

int parent(int a) //find the parent of the given node
{
   while(root[a] != a)
   {
      root[a] = root[root[a]];
a = root[a];
   }
return a;
}

void union_find(int a, int b) //check if the given two vertices are in the same “union” or not
{
   int d = parent(a);
int e = parent(b);
   root[d] = root[e];
}

long long kruskal()
{
   int a, b;
   long long cost, minCost = 0;
for(int i = 0 ; i < edges ; ++i)
   {
      a = p[i].second.first;
b = p[i].second.second;
cost = p[i].first;
      if(parent(a) != parent(b)) //only select edge if it does not create a cycle (ie the two
nodes forming it have different root nodes)
      {
         minCost += cost;
         union_find(a, b);
      }
   }
   return minCost;
}
```

```
int main()
{
   int x, y;
   long long weight, cost, minCost;
   for(int i = 0;i < MAX;++i)                              //initialize the array groups     {
      root[i] = i;
   }
   p[0] = make_pair(10, make_pair(0, 1));
p[1] = make_pair(18, make_pair(1, 2));
  p[2]  = make_pair(13, make_pair(2, 3));
p[3] = make_pair(21, make_pair(0, 2));
p[4] = make_pair(22, make_pair(1, 3));
   sort(p, p + edges);                                  //sort the array of edges
minCost = kruskal();
   cout << "Minimum cost is: "<< minCost << endl;
return 0;
}
```

OUTPUT

```
Minimum cost is: 41
```