```
//Prims algo
#include <iostream>
#include <vector>
using namespace std;

const int INF = INT32_MAX;

void addEdge(vector<vector<pair<int, int>>> &adj, int u, int v, int w)
{
   adj[u].push_back({v, w});
   adj[v].push_back({u, w});
}

int currmindis(int n, vector<int> &distance, vector<bool> &Visited)
{
   int mindistance = INF, currver = -1;
   for (int i = 0; i < n; i++)
   {
      if (!Visited[i] && distance[i] < mindistance)
      {
         mindistance = distance[i];
         currver = i;
      }
   }
   return currver;
}

vector<int> prims(vector<vector<pair<int, int>>> &adj)
{
   int n = adj.size();
   vector<int> distance(n, INF), parent(n, -1);
   vector<bool> Visited(n, false);

   int src = 0;
   distance[src] = 0;

   for (int i = 0; i < n - 1; i++)
   {
      int u = currmindis(n, distance, Visited);
      Visited[u] = true;

      for (auto &edge : adj[u])
      {
         int v = edge.first;
         int w = edge.second;
         if (!Visited[v] && w < distance[v])
         {
            distance[v] = w;
            parent[v] = u;
         }
      }
   }

   return parent;
}

void display(vector<int> &parent)
{
   for (int i = 1; i < parent.size(); i++)
   {
      cout << parent[i] << " - " << i << "\n";
   }
}

int main()
{
   int n, m;
```

```
    cout << "Enter No. of vertices and edges" << endl;
    cin >> n >> m;
    vector<vector<pair<int, int>>> adj(n);

    for (int i = 0; i < m; i++)
    {
        int u, v, w;
        cout << "Enter starting -ending - weight of edge " << i + 1 << endl;
        cin >> u >> v >> w;
        addEdge(adj, u, v, w);
    }

    vector<int> parent = prims(adj);

    display(parent);

    return 0;
}
```

```
PS E:\GIt> cd "e:\GIt\SEM-4\AOA\" ; if ($?) { g++ PrimsAlgo.cpp -o PrimsAlgo } ; if ($?) { .\PrimsAlgo }
Enter No. of vertices and edges
3
3
Enter starting -ending - weight of edge 1
0 1 1
Enter starting -ending - weight of edge 2
1 2 3
Enter starting -ending - weight of edge 3
2 0 2
0 - 1
0 - 2
PS E:\GIt\SEM-4\AOA>
```