ASSIGNMENT NO.12

NAME : PAYAL BHAGVAN CHARVANDE

PRN NO.: 125B2F002

CLASS: SY **DIV:** IT A

CODE:

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

struct Edge {
   int u, v;
   float w;
};
// Find parent (Union-Find)
int findParent(int parent[], int i) {
   if (parent[i] == i)
      return i;
   return parent[i] = findParent(parent, parent[i]);
}
// Union of two sets
void unionSet(int parent[], int rank[], int x, int y) {
   int xroot = findParent(parent, x);
   int yroot = findParent(parent, y);
   if (rank[xroot] < rank[yroot])
      parent[xroot] = yroot;
```

```
    else if (rank[xroot] > rank[yroot])
        parent[yroot] = xroot;
    else {
        parent[yroot] = xroot;
        rank[xroot]++;
    }
}
// Sort edges by cost (Bubble Sort)
void sortEdges(Edge edges[], int E) {
    for (int i = 0; i < E - 1; i++) {
        for (int j = 0; j < E - i - 1; j++) {
            if (edges[j].w > edges[j + 1].w) {
                Edge temp = edges[j];
                edges[j] = edges[j + 1];
                edges[j + 1] = temp;
            }
        }
    }
}
// Kruskal's Algorithm
void kruskalMST(Edge edges[], int V, int E) {
    sortEdges(edges, E);  // Sort edges by cost
  int parent[50], rank[50];
    for (int i = 1; i <= V; i++) {
        parent[i] = i;
        rank[i] = 0;
```

```
    }
    Edge result[50];
    int e = 0;
    float totalCost = 0;
    int i = 0;

    while (e < V - 1 && i < E) {
        Edge next = edges[i++];
        int x = findParent(parent, next.u);
        int y = findParent(parent, next.v);

        if (x != y) {  // No cycle
            result[e++] = next;
            unionSet(parent, rank, x, y);
            totalCost += next.w;
        }
    }
    cout << "\nMinimum Pipeline Network (MST):\n";
    for (int j = 0; j < e; j++) {
        cout << "House " << result[j].u << " - House " << result[j].v
             << " | Cost: " << result[j].w << endl;
    }
    cout << "\nMinimum Total Pipeline Cost = " << totalCost << endl;
}
int main() {
    int V, E;
```

```
    cout << "Enter number of houses (vertices): ";
    cin >> V;
    cout << "Enter number of possible connections (edges): ";
    cin >> E;

    Edge edges[50];

    cout << "\nEnter each connection as: House1 House2 Cost\n";
    for (int i = 0; i < E; i++) {
        cout << "Edge " << i + 1 << ": ";
        cin >> edges[i].u >> edges[i].v >> edges[i].w;
    }
    kruskalMST(edges, V, E);
    return 0;
}
```

OUTPUT:

```
Enter number of houses (vertices): 4
Enter number of possible connections (edges): 5

Enter each connection as: House1 House2 Cost
Edge 1: 1 2 3
Edge 2: 1 3 1
Edge 3: 2 3 7
Edge 4: 2 4 5
Edge 5: 3 4 4

Minimum Pipeline Network (MST):
House 1 - House 3 | Cost: 1
House 1 - House 2 | Cost: 3
House 3 - House 4 | Cost: 4

Minimum Total Pipeline Cost = 8
```