// Kruskals Algorithm

// Note - Graph indices start from 1 not 0

#include <iostream>

#include<string.h>

#define MAX 10

using namespace std;

class kruskal

{

public:

int i, j, k, a, b, u, v, n, ne;

int min, mincost, cost[MAX][MAX], parent[MAX]; //parent variable stores the path of edges

int find(int);

int uni(int, int);

void mainkruskal();

void create();

kruskal() {

ne = 1; //number of edge

mincost = 0; //minimum cost

//initialize cost[][]

for (int i = 1; i <= MAX; i++)

for (int j = 1; j <= MAX; j++)

cost[i][j] = 999;

//initialize parent[]

for (int i = 1; i <= MAX; i++)

parent[i] = 0;

}

};

void kruskal::create()

{

int ch, wt;

cout << "No. of Vertices in a graph :: ";

cin >> n;

for (int i = 1; i < n; i++)//Vertices/Nodes

{

for (int j = i + 1; j <= n; j++)//Adjacency nodes

{

cout << "Does edge is present between " << i << " and " << j << "?(1/0)- ";

cin >> ch;

if (ch == 1) {

cout << " Enter the weight? ";

cin >> wt;

cost[i][j] = cost[j][i] = wt;

}

}//inner for -j

}//outer for - i

}//create

void kruskal::mainkruskal() {

while (ne < n) { //number of edges are less than the vertices

for (i = 1, min = 999; i <= n; i++) {

for (j = 1; j <= n; j++) {

if (cost[i][j] < min) {

min = cost[i][j];

a = u = i;

b = v = j;

}//if

}//inner for

}//outer for

u = find(u); //To find whether vertex u is a part of added edge or not

v = find(v); //To find whether vertex v is a part of added edge or not

if (uni(u, v)) { //If no cycle

cout << "edge " << ne++ << " (" << a << "," << b << ") " << min << "\n";

mincost += min;

}//if

cost[a][b] = cost[b][a] = 999; //not to be considered again

}//while

cout << "\ntMinimum cost = " << mincost << "\n";

}//kruskals

int kruskal::find(int i)

{

while (parent[i])

i = parent[i]; //For the path of added vertices

return i;

}//find

int kruskal::uni(int i, int j)

{

if (i != j) //If no cycle

{

parent[j] = i;

return 1;

}

return 0;

}//union

int main()

{

kruskal obj; //constructor will be called and initialization are made

obj.create();

obj.mainkruskal();

return 0;

}