#include <cstdlib>

#include <iostream>

#define inf 9999

using namespace std;

class prims {

public:

int cost[10][10], vertex, edge;

void creategraph();

void primfun(int);

};

void prims::creategraph() {

int v1, v2, i, j, wt;

cout << "\n\t\t Enter the MAXIMUM no of offices:::";

cin >> vertex;

cout << "\n\t\t Enter the no of lease lines:::";

cin >> edge;

// initialize cost matrix

for (i = 0; i < vertex; i++)

for (j = 0; j < vertex; j++) {

if (i == j)

cost[i][j] = 0;

else

cost[i][j] = inf;

}

//enter information about lease lines

for (i = 0; i < edge; i++) {

cout << "\n\t\t Enter the lease lines (edge) and Their costs(v1,v2,wt)::";

cin >> v1 >> v2 >> wt;

cost[v1][v2] = cost[v2][v1] = wt;

}

}

void prims::primfun(int s) {

int min, i, j, n = 1, visited[10], dist[10], from[10], nextnode, mstcost = 0;

//initialization

for (i = 0; i < vertex; i++) {

visited[i] = 0;

dist[i] = inf;

from[i] = s;

}

//start node visited

visited[s] = 1;

//update distance array

for (i = 0; i < vertex; i++) {

if (visited[i] == 0 && cost[s][i] < dist[i]) {

dist[i] = cost[s][i];

}

}

//for vertex-1 no of edges

while (n < vertex) {

min = inf;

//find next node to visit

for (i = 0; i < vertex; i++) {

if (visited[i] == 0 && dist[i] < min) {

min = dist[i];

nextnode = i;

}

}

cout << endl << from[nextnode] << " " << nextnode << endl;

n++;

visited[nextnode] = 1;

mstcost += dist[nextnode];

//update distance array

for (i = 0; i < vertex; i++) {

if (visited[i] == 0 && dist[i] > cost[nextnode][i]) {

dist[i] = cost[i][nextnode];

from[i] = nextnode;

}

}

}

cout << "\n\tCost of minimum spanning tree is::" << mstcost << endl;

}

int main() {

prims s1;

int ch;

while (1) {

cout << "\n 1.Creategraph (Adjacency matrix form) \n 2.Prims Algorithm \n 3.Exit.\n";

cout << "\n\nEnter Ur Choice= ";

cin >> ch;

switch (ch) {

case 1:

s1.creategraph();

break;

case 2:

int start;

cout << "\nEnter Starting Vertex=";

cin >> start;

cout << endl << "MST is: " << endl;

s1.primfun(start);

break;

case 3:

exit(0);

}

}

return 0;

}