// Prim's Algorithm in C++

// The time complexity of Prim's algorithm is O(E log V).

#include <cstring>

#include <iostream>

using namespace std;

#define INF 9999999

// number of vertices in grapj

#define V 50

// create a 2d array of size 5x5

// for adjacency matrix to represent graph

class Graph{

public:

int G[V][V];

};

// int G[V][V] = {

// {0, 9, 75, 0, 0},

// {9, 0, 95, 19, 42},

// {75, 95, 0, 51, 66},

// {0, 19, 51, 0, 31},

// {0, 42, 66, 31, 0}};

int main()

{

int ver;

cout<<"Enter number of vertices : ";

cin>>ver;

Graph g;

for(int i=0;i<ver;i++){

for(int j=0;j<ver;j++){

cout<<"Edge weight "<<i<<" "<<j<<" : ";

cin>>g.G[i][j];

}

}

int no\_edge; // number of edge

// create a array to track selected vertex

// selected will become true otherwise false

int selected[ver];

// set selected false initially

for (int i = 0; i < ver; i++)

{

selected[i] = false;

}

// set number of edge to 0

no\_edge = 0;

// the number of egde in minimum spanning tree will be

// always less than (V -1), where V is number of vertices in

// graph

// choose 0th vertex and make it true

selected[0] = true;

int x; // row number

int y; // col number

// print for edge and weight

cout << "Edge"

<< " : "

<< "Weight";

cout << endl;

while (no\_edge < ver - 1)

{

// For every vertex in the set S, find the all adjacent vertices

// , calculate the distance from the vertex selected at step 1.

// if the vertex is already in the set S, discard it otherwise

// choose another vertex nearest to selected vertex at step 1.

int min = INF;

x = 0;

y = 0;

for (int i = 0; i < ver; i++)

{

if (selected[i])

{

for (int j = 0; j < ver; j++)

{

if (!selected[j] && g.G[i][j])

{ // not in selected and there is an edge

if (min > g.G[i][j])

{

min = g.G[i][j];

x = i;

y = j;

}

}

}

}

}

cout << x << " - " << y << " : " << g.G[x][y];

cout << endl;

selected[y] = true;

no\_edge++;

}

return 0;

}