#include <iostream>

using namespace std;

const int MAX\_VERTICES = 100;

int cost[MAX\_VERTICES][MAX\_VERTICES], parent[MAX\_VERTICES];

int n;

void find\_min(int& x, int& y);

int check\_cycle(int x, int y);

int main() {

int count = 0, tot = 0, flag = 0;

clock\_t start,end;

double clk;

cout << "Enter the number of vertices: ";

cin >> n;

cout << "Enter the cost matrix (Enter 999 for No edges and self-loops):\n";

for (int i = 1; i <= n; ++i) {

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

cin >> cost[i][j];

parent[j] = 0;

}

}

start=clock();

while (count != n - 1) {

int x, y;

find\_min(x, y);

flag = check\_cycle(x, y);

if (flag == 1) {

cout << "\n" << x << " -----> " << y << " == " << cost[x][y] << "\n";

count++;

tot += cost[x][y];

}

cost[x][y] = cost[y][x] = 999;

}

end=clock();

cout << "\nThe total cost of the minimum spanning tree = " << tot << endl;

clk=(double)(end-start)/CLOCKS\_PER\_SEC;

cout<<"The run time is"<<clk<<"sec";

return 0;

}

void find\_min(int& x, int& y) {

int min\_val = 999;

for (int i = 1; i <= n; ++i) {

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

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

min\_val = cost[i][j];

x = i;

y = j;

}

}

}

}

int check\_cycle(int x, int y) {

if ((parent[x] == parent[y]) && (parent[x] != 0))

return 0;

else if (parent[x] == 0 && parent[y] == 0)

parent[x] = parent[y] = x;

else if (parent[x] == 0)

parent[x] = parent[y];

else if (parent[x] != parent[y])

parent[y] = parent[x];

return 1;

}