#include <stdio.h>

#define INF 99999

#define MAX 100

int graph[MAX][MAX]; // Adjacency matrix

int visited[MAX]; // Visited vertices

// Function to find the vertex with minimum edge cost

int findMinVertex(int key[], int visited[], int n) {

int min = INF, minIndex = -1;

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

if (!visited[i] && key[i] < min) {

min = key[i];

minIndex = i;

}

}

return minIndex;

}

// Prim's Algorithm

void primMST(int n) {

int parent[MAX]; // Stores MST structure

int key[MAX]; // Minimum edge weight to connect

int cost = 0;

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

key[i] = INF;

visited[i] = 0;

}

key[0] = 0; // Start from vertex 0

parent[0] = -1; // Root of MST

for (int count = 0; count < n - 1; count++) {

int u = findMinVertex(key, visited, n);

visited[u] = 1;

for (int v = 0; v < n; v++) {

if (graph[u][v] && !visited[v] && graph[u][v] < key[v]) {

parent[v] = u;

key[v] = graph[u][v];

}

}

}

// Print MST edges and total cost

printf("Edges in Minimum Spanning Tree:\n");

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

printf("%d - %d : %d\n", parent[i], i, graph[i][parent[i]]);

cost += graph[i][parent[i]];

}

printf("Total cost of MST: %d\n", cost);

}

int main() {

int n;

// Input: number of vertices

printf("Enter number of vertices: ");

scanf("%d", &n);

// Input: adjacency matrix

printf("Enter adjacency matrix (0 if no edge):\n");

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

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

scanf("%d", &graph[i][j]);

// Run Prim's Algorithm

primMST(n);

return 0;

}