```
#include <iostream>

#include <climits>

using namespace std;



const int MAX = 100;



int findMinKey(int key[], bool inMST[], int n) {

int minIndex = -1, minValue = INT_MAX;

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

if (!inMST[i] && key[i] < minValue) {

minValue = key[i];

minIndex = i;

}

}

return minIndex;

}
```

```
void primMST(int graph[MAX][MAX], int n) {

int parent[MAX], key[MAX];

bool inMST[MAX] = {false};



fill(key, key + n, INT_MAX);

key[0] = 0;

parent[0] = -1;



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

int u = findMinKey(key, inMST, n);

inMST[u] = true;



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

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

parent[v] = u;

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

}
}

}

cout << "Edge\tWeight\n";

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

cout << parent[i] << " - " << i << "\t" << graph[i][parent[i]] << "\n";

}

int main() {

int n, graph[MAX][MAX];

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

cin >> n;

cout << "Enter adjacency matrix (0 for no edge):\n";

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

cin >> graph[i][j];




cout << "\nMinimum Spanning Tree (MST):\n";

primMST(graph, n);




return 0;

}
```