// spanning tree for adjacency representation.

#include <bits/stdc++.h>

using namespace std;

#define V 5

// Returns true if edge u-v is a valid edge to be include in MST.

//An edge is valid if one end is already included in MST and other is not in MST.

bool isValidEdge(int u, int v, vector<bool> inMST)

{

if (u == v)

return false;

if (inMST[u] == false && inMST[v] == false)

return false;

else if (inMST[u] == true && inMST[v] == true)

return false;

return true;

}

void primMST(int cost[][V])

{

vector<bool> inMST(V, false);

// Include first vertex in MST

inMST[0] = true;

// Keep adding edges while number of included

// edges does not become V-1.

int edge\_count = 0, mincost = 0;

while (edge\_count < V - 1) {

// Find minimum weight valid edge.

int min = INT\_MAX, a = -1, b = -1;

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

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

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

if (isValidEdge(i, j, inMST)) {

min = cost[i][j];

a = i;

b = j;

}

}

}

}

if (a != -1 && b != -1) {

printf("Edge %d:(%d, %d) cost: %d \n",

edge\_count++, a, b, min);

mincost = mincost + min;

inMST[b] = inMST[a] = true;

}

}

printf("\n Minimum cost= %d \n", mincost);

}

// driver program to test above function

int main()

{

/\* Let us create the following graph

2 3

(0)--(1)--(2)

| / \ |

6| 8/ \5 |7

| / \ |

(3)-------(4)

9 \*/

int cost[][V] = {

{ INT\_MAX, 2, INT\_MAX, 6, INT\_MAX },

{ 2, INT\_MAX, 3, 8, 5 },

{ INT\_MAX, 3, INT\_MAX, INT\_MAX, 7 },

{ 6, 8, INT\_MAX, INT\_MAX, 9 },

{ INT\_MAX, 5, 7, 9, INT\_MAX },

};

// Print the solution

primMST(cost);

return 0;

}