Name: Sakshi Mishra

Roll:53

Subject:DSA

LAB ASSIGNMENT NO. 07

```
// A C++ program for Prim's Minimum
// Spanning Tree (MST) algorithm. The program is
// for adjacency matrix representation of the graph
#include <bits/stdc++.h> using
namespace std;
// Number of vertices in the graph
#define V 5
// A utility function to find the vertex with
// minimum key value, from the set of vertices
// not yet included in MST int
minKey(int key[], bool mstSet[])
{
// Initialize min value int min =
INT_MAX, min_index;
for (int i = 0; i < V; i++) if (mstSet[i]
== false && key[i] < min) min =
key[i], min_index = i; return
min_index;
}
// A utility function to print the // constructed
MST stored in parent[] void printMST(int
parent[], int graph[V][V])
{
cout << "Edge \tWeight\n"; for (int i
= 1 ;i < V; i++ ) cout << parent[i] <<
" - " << i << " \t" <<
graph[i][parent[i]] << " \n";
```

```
}
// Function to construct and print MST for
// a graph represented using adjacency
// matrix representation void
primMST(int graph[V][V])
{
// Array to store constructed MST int
parent[V];
// Key values used to pick minimum weight edge in cut int
key[V];
// To represent set of vertices included in MST
bool mstSet[V]; // Initialize all keys as
INFINITE for (int i = 0; i < V; i++) key[i] =
INT_MAX, mstSet[i] = false;
// Always include first 1st vertex in MST. //
Make key 0 so that this vertex is picked as first //
vertex. key[0] = 0;
// First node is always root of MST
parent[0] = -1; // The MST will have V
vertices for (int count = 0; count < V - 1;
count++) { // Pick the minimum key vertex
from the // set of vertices not yet included
in MST int u = minKey(key, mstSet);

// Add the picked vertex to the MST Set mstSet[u]
= true;
// Update key value and parent index of
// the adjacent vertices of the picked vertex.
// Consider only those vertices which are not
// yet included in MST for
(int v = 0; v < V; v++)
// graph[u][v] is non zero only for adjacent vertices of m
```

```
// mstSet[v] is false for vertices not yet included in MST //
Update the key only if graph[u][v] is smaller than key[v] if
(graph[u][v] && mstSet[v] == false && graph[u][v] <
key[v]) parent[v] = u, key[v] = graph[u][v];
}
// Print the constructed MST printMST(parent,
graph);
} // Driver's
code int main()
{
int graph[V][V] = { { 0, 2, 0, 6, 0 },
{ 2, 0, 3, 8, 5 },
{ 0, 3, 0, 0, 7 },
{ 6, 8, 0, 0, 9 },
{ 0, 5, 7, 9, 0 } }; //
Print the solution
primMST(graph);
return 0;
}
```

Output

```
Edge  Weight
0 - 1     2
1 - 2     3
0 - 3     6
1 - 4     5
```