CSA 0317 DATA STRUCTURES

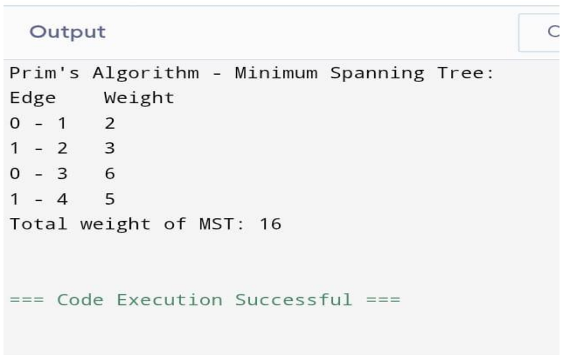
PROGRAM 24

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

#define V 5  // Number of vertices

// Function to find the vertex with minimum key value
int minKey(int key[], bool mstSet[]) {
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++) {
        if (mstSet[v] == false && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    }
    return min_index;
}

// Function to print the constructed MST
void printMST(int parent[], int graph[V][V]) {
    printf("Edge \tWeight\n");
    int totalWeight = 0;
    for (int i = 1; i < V; i++) {
        printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
        totalWeight += graph[i][parent[i]];
    }
    printf("Total weight of MST: %d\n", totalWeight);
```

```
}

// Prim's algorithm
void primMST(int graph[V][V]) {
  int parent[V];  // Array to store constructed MST
  int key[V];     // Key values to pick minimum weight edge
  bool mstSet[V]; // To represent set of vertices included in MST

  // Initialize all keys as INFINITE
  for (int i = 0; i < V; i++) {
    key[i] = INT_MAX;
    mstSet[i] = false;
  }

  // Always include first vertex in MST
  key[0] = 0;     // Make key 0 so this vertex is picked first
  parent[0] = -1; // First node is always root of MST

  // The MST will have V vertices
  for (int count = 0; count < V - 1; count++) {
    // Pick the minimum key vertex not yet included in MST
    int u = minKey(key, mstSet);

    // Add the picked vertex to MST set
    mstSet[u] = true;

    // Update key value and parent index of adjacent vertices
    for (int v = 0; v < V; v++) {
      if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) {
        parent[v] = u;
        key[v] = graph[u][v];
```

```
            }
        }
    }

    printMST(parent, graph);
}

int main() {
    // Example graph represented as adjacency matrix
    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}
    };

    printf("Prim's Algorithm - Minimum Spanning Tree:\n");
    primMST(graph);

    return 0;
}
```

Output:



```
Prim's Algorithm - Minimum Spanning Tree:
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
Total weight of MST: 16


=== Code Execution Successful ===
```