S.No: 5	Exp. Name: ***Prims Algorithm***	Date:2025-08-07

ID: 24070521030

Aim:

Write a C program to implement Prim's algorithm for finding the Minimum Cost Spanning Tree of a given undirected graph represented by an adjacency matrix.

Input Format:

- The first line contains an integer n, representing the number of vertices in the graph.
- The next n lines each contain n space-separated integers, representing the adjacency matrix of the undirected weighted graph.
- The value at row i and column j denotes the weight of the edge between vertex i and vertex j.
- A value of "0" indicates that there is no edge between the corresponding vertices.

Output Format:

- The program prints the Minimum Spanning Tree (MST) as edges along with their weights.

Note:

- The algorithm starts from **vertex 0**.
- Refer to the visible test cases for better understanding.

Source Code:

minCostFinding.c

```
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#define V 100

int minKey(int key[], bool mstSet[], int vertices) {
   int min = INT_MAX,min_index;
   for(int v = 0 ; v < vertices; v++)
      {
         if(!mstSet[v] && key[v] < min)
         {
            min= key[v];
            min_index = v;
         }
      }
         return min_index;

}
void printTree(int parent[], int graph[V][V], int vertices) {
    printf("Edge \tWeight\n");
    for (int i = 1; i < vertices; i++)
        printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
}

void prim(int graph[V][V], int vertices) {
   int parent[V];
   int key[V];
   bool mstSet[V];
   for(int i = 0 ; i< vertices; i++)
```

```
      {
         key[i] = INT_MAX;
         mstSet[i]= false;
      }
   key[0] = 0;
   parent[0] = -1;
   for(int count = 0; count < vertices-1; count++){
      int u = minKey(key, mstSet, vertices);
      mstSet[u] = true;
      for(int v = 0 ; v < vertices; v++)
         {
            if(graph[u][v] && !mstSet[v] && graph[u][v])
            {
               parent[v] =u;
               key[v]= graph[u][v];
            }
         }
   }
   printTree(parent , graph, vertices);
}

int main() {
    int vertices;
    int graph[V][V];

    printf("No of vertices: ");
    scanf("%d", &vertices);

    printf("Adjacency matrix elements (row wise):\n");
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    prim(graph, vertices);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
No of vertices: 5
Adjacency matrix elements (row wise): 0 0 4 0 0
0 0 5 3 0
4 5 0 0 0
0 3 0 0 2
0 0 0 2 0
Edge Weight
2 - 1 5
0 - 2 4

ID: 24070521030

1 - 3	3
3 - 4	2