Pract 3

```
import java.util.Arrays;

public class Main {

   private static final int V = 5; // Number of vertices

   // Function to find the vertex with the minimum key value
   int minKey(int key[], Boolean mstSet[]) {
      int min = Integer.MAX_VALUE, min_index = -1;
      for (int v = 0; v < V; v++)
         if (!mstSet[v] && key[v] < min) {
            min = key[v];
            min_index = v;
         }
      return min_index;
   }

   // Function to print the MST
   void printMST(int parent[], int graph[][]) {
      System.out.println("Edge \tWeight");
      for (int i = 1; i < V; i++)
         System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
   }

   // Function to find the MST using Prim's algorithm
   void primMST(int graph[][]) {
      int parent[] = new int[V]; // Array to store the constructed MST
      int key[] = new int[V]; // Key values used to pick the minimum weight edge in each cut
      Boolean mstSet[] = new Boolean[V]; // To represent set of vertices included in MST
      for (int i = 0; i < V; i++) {
         key[i] = Integer.MAX_VALUE; // Initialize key values to infinity
         mstSet[i] = false; // Initially no vertices are included in MST
      }
      key[0] = 0; // Key value of the first vertex is 0
      parent[0] = -1; // First vertex is always root of MST

      // Construct MST with V-1 edges
      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);
         mstSet[u] = true; // Include the picked vertex in MST

         // Update key values and parent index of the adjacent vertices of the picked vertex
         for (int v = 0; v < V; v++)
            if (graph[u][v] != 0 && !mstSet[v] && graph[u][v] < key[v]) {
               parent[v] = u;
               key[v] = graph[u][v];
            }
      }
      // Print the constructed MST
      printMST(parent, graph);
   }

   // Main method
   public static void main(String[] args) {
```

```
        Main t = new Main();
        // Input graph represented as an adjacency matrix
        int graph[][] = new int[][]{{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}};
        // Find and print the MST of the input graph
        t.primMST(graph);
    }
}
```