Name: Vijay Misal
Div: C Batch: C3
Roll No: 233073
PRN No: 22320079

Practical No: 5

Title: Use the map of the area around the college as the graph. Identify the prominent land marks as nodes and find minimum distance to various land marks from the college as the source. Represent this graph using adjacency matrix. Find the shortest path using Dijkstra's algorithm.

Code:

```java
import java.util.Scanner;

public class Practical5 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of landmarks: ");
        int n = scanner.nextInt(); // number of landmarks
        int[][] graph = new int[n][n]; // adjacency matrix to represent the graph
        System.out.println("Enter the adjacency matrix: ");
        for (int i = 0; i < n; i++) { // input the adjacency matrix
            for (int j = 0; j < n; j++) {
                graph[i][j] = scanner.nextInt();
            }
        }
        System.out.println("Enter the source landmark: ");
        int source = scanner.nextInt(); // source landmark
        dijkstra(graph, source); // find the shortest path using Dijkstra's algorithm
        scanner.close();
    }

    public static void dijkstra(int[][] graph, int source) {
        int n = graph.length; // number of landmarks
        int[] distance = new int[n]; // array to store the distance from source landmark to
other landmarks
        boolean[] visited = new boolean[n]; // array to store the visited landmarks
        for (int i = 0; i < n; i++) {
            distance[i] = Integer.MAX_VALUE; // initialize the distance array and visited array
            visited[i] = false;
        }
        distance[source] = 0; // distance from source to source is 0
        for (int i = 0; i < n - 1; i++) { // find the shortest path to all landmarks
            int u = minDistance(distance, visited); // find the landmark with minimum distance
            visited[u] = true; // mark the landmark as visited
            for (int v = 0; v < n; v++) { // update the distance array for the adjacent
landmarks
                if (!visited[v] && graph[u][v] != 0 && distance[u] != Integer.MAX_VALUE // if
the landmark is not
                                                                                        //
visited and there is an edge
                                                                                        // from
u to v and the distance
```

```
                                                                                // from
source to u is not
                                                                                //
infinity
                        && distance[u] + graph[u][v] < distance[v]) {
                    distance[v] = distance[u] + graph[u][v];
                }
            }
        }
        printSolution(distance);
    }

    public static int minDistance(int[] distance, boolean[] visited) {
        int min = Integer.MAX_VALUE;
        int minIndex = -1;
        for (int i = 0; i < distance.length; i++) {
            if (!visited[i] && distance[i] <= min) {
                min = distance[i];
                minIndex = i;
            }
        }
        return minIndex;
    }

    public static void printSolution(int[] distance) {
        System.out.println("Landmark \t Distance from source");
        for (int i = 0; i < distance.length; i++) {
            System.out.println(i + " \t\t " + distance[i]);
        }
    }

}
```

Output:

Enter the number of landmarks:

4

Enter the adjacency matrix:

12

24

45

56

23

45

76

44

34

22

88

65

45

33

54

22

Enter the source landmark:

3

Landmark	Distance from source
0	45
1	33
2	54
3	0