Name: Samir Kumar
USN: 1BM18CS091

Page No.
Date: / /

* Implement Dijkstra's algorithm to compute the shortest path through a graph.

```
import java.util.*;

class Edge {
    int src, dest, w;
    public Edge(int src, int dest, int w){
        this.src = src;
        this.dest = dest;
        this.w = w;
    } }

class Node {
    int vertex, w;
    public Node(int vertex, int w){
        this.vertex = vertex;
        this.w = w;
    } }

class graph{
    List<List<Edge>> edgelist = null;
    graph(List<Edge> edges, int N){

    edgelist = new ArrayList<>();
    for(int i = 0; i<N; i++) {
        edgelist.add(new ArrayList<>());
    }
```

```
        for(Edge edge : edges){
            edgelist.get(edge.src).add(edge);
        }
    }
}

class Dijkstra{
    private static void getPath(int[] prev, int i,
            List<Integer> route){
        if(i>=0){
            getPath(prev, prev[i], route);
            route.add(i);
        }
    }

public static void getShortestPath(Graph graph,
    int src, int N){
    PriorityQueue<Node> minHeap;

minHeap=new PriorityQueue<>(Comparator.comparing(n
                (n -> n.w));

minheap.add(new Node(src,0));
List<Integer> dist= new ArrayList<>(Collections.
        nCopies(N, Integer.MAX_VALUE));
dist.set(src,0);
boolean[] done = new boolean[N];
```

done [src] = true;

int [] prev = new int [N];

prev [src] = -1

List<Integer> route = new ArrayList<>();

while (!minHeap.isEmpty()){

Node node = minHeap.poll();

done [u] =

true;

}

for(int i = 1; i < N ; ++i){

if (i != src && dist.get(i) != Integer.MAX_VALUE){

getPath (prev, i route)

route.clear();

}

}

```
public static void main(String[] args){
    Scanner s = new Scanner(System.in);
    List<Edge> edges = new ArrayList<>();
    System.out.println("Enter no. of vertices");
    int n = s.nextInt();
    int[][] mat = new int[n][n];
    mat[i][j] = s.nextInt();
  }
}

for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        if(i==j) continue;
        if(mat[i][j] != -1){
        edges.add(new Edge(i,j, mat[i][j]));
}}}

Graph graph = new Graph(edges, n);
int src = 0;
getShortestPath(graph, src, n);
s.close();
}}
```