NAME : RASAVE PRALHAD MAROTI
ROLL NO : 55

ASS NO : 2

```
import java.util.*;


class AStar {
   private static final int[][] MOVES = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};


   private static int heuristic(int x, int y, int goalX, int goalY) {
      return Math.abs(x - goalX) + Math.abs(y - goalY);
   }


   public static List<int[]> findPath(int[][] grid, int[] start, int[] goal) {
      int rows = grid.length;
      int cols = grid[0].length;


      PriorityQueue<Node> openList = new PriorityQueue<>(Comparator.comparingInt(n -> n.f));
      boolean[][] closedSet = new boolean[rows][cols];


      Node startNode = new Node(start[0], start[1], 0, heuristic(start[0], start[1], goal[0], goal[1]),
null);
      openList.add(startNode);


      while (!openList.isEmpty()) {
         Node currentNode = openList.poll();
         int x = currentNode.x;
         int y = currentNode.y;


         if (x == goal[0] && y == goal[1]) {
            return reconstructPath(currentNode);
         }
```

```
            closedSet[x][y] = true;

            for (int[] move : MOVES) {
                int newX = x + move[0];
                int newY = y + move[1];

                if (newX >= 0 && newX < rows && newY >= 0 && newY < cols && grid[newX][newY] == 1
&& !closedSet[newX][newY]) {
                    int g = currentNode.g + 1;
                    int h = heuristic(newX, newY, goal[0], goal[1]);
                    Node neighbor = new Node(newX, newY, g, h, currentNode);
                    openList.add(neighbor);
                }
            }
        }

        return new ArrayList<>();
    }

    private static List<int[]> reconstructPath(Node node) {
        List<int[]> path = new ArrayList<>();
        while (node != null) {
            path.add(new int[]{node.x, node.y});
            node = node.parent;
        }
        Collections.reverse(path);
        return path;
    }

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

```
        int[][] grid = {
            {1, 1, 1, 1, 1},
            {1, 0, 0, 0, 1},
            {1, 1, 1, 0, 1},
            {1, 1, 1, 1, 1}
        };

        int[] start = {0, 0};
        int[] goal = {3, 4};

        List<int[]> path = findPath(grid, start, goal);

        if (!path.isEmpty()) {
            System.out.println("Path found:");
            for (int[] p : path) {
                System.out.println(Arrays.toString(p));
            }
        } else {
            System.out.println("No path found.");
        }
    }
}

class Node {
    int x, y;
    int g;
    int h;
    int f;
    Node parent;

    public Node(int x, int y, int g, int h, Node parent) {
```

```
        this.x = x;
        this.y = y;
        this.g = g;
        this.h = h;
        this.f = g + h;
        this.parent = parent;
    }
}
```

OUTPUT :

```
Path found:
[0, 0]
[1, 0]
[2, 0]
[2, 1]
[2, 2]
[3, 2]
[3, 3]
[3, 4]

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