import java.util.\*;

import java.util.HashMap;

import java.util.Map;

import java.util.LinkedHashMap;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

class Cell {

    int parent\_i, parent\_j;

    double f, g, h;

    Cell() {

        this.parent\_i = 0;

        this.parent\_j = 0;

        this.f = 0;

        this.g = 0;

        this.h = 0;

    }

}

public class AStarSearch {

    private static final int ROW = 9;

    private static final int COL = 10;

    public static void main(String[] args) {

        // Description of the Grid-

        // 1--> The cell is not blocked

        // 0--> The cell is blocked

        int[][] grid = {

                {1, 0, 1, 1, 1, 1, 0, 1, 1, 1},

                {1, 1, 1, 0, 1, 1, 1, 0, 1, 1},

                {1, 1, 1, 0, 1, 1, 0, 1, 0, 1},

                {0, 0, 1, 0, 1, 0, 0, 0, 0, 1},

                {1, 1, 1, 0, 1, 1, 1, 0, 1, 0},

                {1, 0, 1, 1, 1, 1, 0, 1, 0, 0},

                {1, 0, 0, 0, 0, 1, 0, 0, 0, 1},

                {1, 0, 1, 1, 1, 1, 0, 1, 1, 1},

                {1, 1, 1, 0, 0, 0, 1, 0, 0, 1}

        };

        // Source is the left-most bottom-most corner

        int[] src = {8, 0};

        // Destination is the left-most top-most corner

        int[] dest = {0, 0};

        aStarSearch(grid, src, dest);

    }

    private static boolean isValid(int row, int col) {

        return (row >= 0) && (row < ROW) && (col >= 0) && (col < COL);

    }

    private static boolean isUnBlocked(int[][] grid, int row, int col) {

        return grid[row][col] == 1;

    }

    private static boolean isDestination(int row, int col, int[] dest) {

        return row == dest[0] && col == dest[1];

    }

    private static double calculateHValue(int row, int col, int[] dest) {

        return Math.sqrt((row - dest[0]) \* (row - dest[0]) + (col - dest[1]) \* (col - dest[1]));

    }

    private static void tracePath(Cell[][] cellDetails, int[] dest) {

        System.out.println("The Path is ");

        int row = dest[0];

        int col = dest[1];

        Map<int[], Boolean> path = new LinkedHashMap<>();

        while (!(cellDetails[row][col].parent\_i == row && cellDetails[row][col].parent\_j == col)) {

            path.put(new int[]{row, col}, true);

            int temp\_row = cellDetails[row][col].parent\_i;

            int temp\_col = cellDetails[row][col].parent\_j;

            row = temp\_row;

            col = temp\_col;

        }

        path.put(new int[]{row, col}, true);

        List<int[]> pathList = new ArrayList<>(path.keySet());

        Collections.reverse(pathList);

        pathList.forEach(p -> {

            if (p[0] == 2 || p[0] == 1) {

                System.out.print("-> (" + p[0] + ", " + (p[1]) + ")");

            } else {

                System.out.print("-> (" + p[0] + ", " + p[1] + ")");

            }

        });

        System.out.println();

    }

    private static void aStarSearch(int[][] grid, int[] src, int[] dest) {

        if (!isValid(src[0], src[1]) || !isValid(dest[0], dest[1])) {

            System.out.println("Source or destination is invalid");

            return;

        }

        if (!isUnBlocked(grid, src[0], src[1]) || !isUnBlocked(grid, dest[0], dest[1])) {

            System.out.println("Source or the destination is blocked");

            return;

        }

        if (isDestination(src[0], src[1], dest)) {

            System.out.println("We are already at the destination");

            return;

        }

        boolean[][] closedList = new boolean[ROW][COL];

        Cell[][] cellDetails = new Cell[ROW][COL];

        for (int i = 0; i < ROW; i++) {

            for (int j = 0; j < COL; j++) {

                cellDetails[i][j] = new Cell();

                cellDetails[i][j].f = Double.POSITIVE\_INFINITY;

                cellDetails[i][j].g = Double.POSITIVE\_INFINITY;

                cellDetails[i][j].h = Double.POSITIVE\_INFINITY;

                cellDetails[i][j].parent\_i = -1;

                cellDetails[i][j].parent\_j = -1;

            }

        }

        int i = src[0], j = src[1];

        cellDetails[i][j].f = 0;

        cellDetails[i][j].g = 0;

        cellDetails[i][j].h = 0;

        cellDetails[i][j].parent\_i = i;

        cellDetails[i][j].parent\_j = j;

        Map<Double, int[]> openList = new HashMap<>();

        openList.put(0.0, new int[]{i, j});

        boolean foundDest = false;

        while (!openList.isEmpty()) {

            Map.Entry<Double, int[]> p = openList.entrySet().iterator().next();

            openList.remove(p.getKey());

            i = p.getValue()[0];

            j = p.getValue()[1];

            closedList[i][j] = true;

            double gNew, hNew, fNew;

            // 1st Successor (North)

            if (isValid(i - 1, j)) {

                if (isDestination(i - 1, j, dest)) {

                    cellDetails[i - 1][j].parent\_i = i;

                    cellDetails[i - 1][j].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i - 1][j] && isUnBlocked(grid, i - 1, j)) {

                    gNew = cellDetails[i][j].g + 1;

                    hNew = calculateHValue(i - 1, j, dest);

                    fNew = gNew + hNew;

                    if (cellDetails[i - 1][j].f == Double.POSITIVE\_INFINITY

 || cellDetails[i - 1][j].f > fNew) {

                        openList.put(fNew, new int[]{i - 1, j});

                        cellDetails[i - 1][j].f = fNew;

                        cellDetails[i - 1][j].g = gNew;

                        cellDetails[i - 1][j].h = hNew;

                        cellDetails[i - 1][j].parent\_i = i;

                        cellDetails[i - 1][j].parent\_j = j;

                    }

                }

            }

            // 2nd Successor (South)

            if (isValid(i + 1, j)) {

                if (isDestination(i + 1, j, dest)) {

                    cellDetails[i + 1][j].parent\_i = i;

                    cellDetails[i + 1][j].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i + 1][j] && isUnBlocked(grid, i + 1, j)) {

                    gNew = cellDetails[i][j].g + 1;

                    hNew = calculateHValue(i + 1, j, dest);

                    fNew = gNew + hNew;

                    if (cellDetails[i + 1][j].f == Double.POSITIVE\_INFINITY || cellDetails[i + 1][j].f > fNew) {

                        openList.put(fNew, new int[]{i + 1, j});

                        cellDetails[i + 1][j].f = fNew;

                        cellDetails[i + 1][j].g = gNew;

                        cellDetails[i + 1][j].h = hNew;

                        cellDetails[i + 1][j].parent\_i = i;

                        cellDetails[i + 1][j].parent\_j = j;

                    }

                }

            }

            // 3rd Successor (East)

            if (isValid(i, j + 1)) {

                if (isDestination(i, j + 1, dest)) {

                    cellDetails[i][j + 1].parent\_i = i;

                    cellDetails[i][j + 1].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i][j + 1] && isUnBlocked(grid, i, j + 1)) {

                    gNew = cellDetails[i][j].g + 1;

                    hNew = calculateHValue(i, j + 1, dest);

                    fNew = gNew + hNew;

                    if (cellDetails[i][j + 1].f == Double.POSITIVE\_INFINITY || cellDetails[i][j + 1].f > fNew) {

                        openList.put(fNew, new int[]{i, j + 1});

                        cellDetails[i][j + 1].f = fNew;

                        cellDetails[i][j + 1].g = gNew;

                        cellDetails[i][j + 1].h = hNew;

                        cellDetails[i][j + 1].parent\_i = i;

                        cellDetails[i][j + 1].parent\_j = j;

                    }

                }

            }

            // 4th Successor (West)

            if (isValid(i, j - 1)) {

                if (isDestination(i, j - 1, dest)) {

                    cellDetails[i][j - 1].parent\_i = i;

                    cellDetails[i][j - 1].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i][j - 1] && isUnBlocked(grid, i, j - 1)) {

                    gNew = cellDetails[i][j].g + 1;

                    hNew = calculateHValue(i, j - 1, dest);

                    fNew = gNew + hNew;

                    if (cellDetails[i][j - 1].f == Double.POSITIVE\_INFINITY || cellDetails[i][j - 1].f > fNew) {

                        openList.put(fNew, new int[]{i, j - 1});

                        cellDetails[i][j - 1].f = fNew;

                        cellDetails[i][j - 1].g = gNew;

                        cellDetails[i][j - 1].h = hNew;

                        cellDetails[i][j - 1].parent\_i = i;

                        cellDetails[i][j - 1].parent\_j = j;

                    }

                }

            }

            // 5th Successor (North-East)

            if (isValid(i - 1, j + 1)) {

                if (isDestination(i - 1, j + 1, dest)) {

                    cellDetails[i - 1][j + 1].parent\_i = i;

                    cellDetails[i - 1][j + 1].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i - 1][j + 1] && isUnBlocked(grid, i - 1, j + 1)) {

                    gNew = cellDetails[i][j].g + 1.414;

                    hNew = calculateHValue(i - 1, j + 1, dest);

                    fNew = gNew + hNew;

                    if (cellDetails[i - 1][j + 1].f == Double.POSITIVE\_INFINITY || cellDetails[i - 1][j + 1].f > fNew) {

                        openList.put(fNew, new int[]{i - 1, j + 1});

                        cellDetails[i - 1][j + 1].f = fNew;

                        cellDetails[i - 1][j + 1].g = gNew;

                        cellDetails[i - 1][j + 1].h = hNew;

                        cellDetails[i - 1][j + 1].parent\_i = i;

                        cellDetails[i - 1][j + 1].parent\_j = j;

                    }

                }

            }

            // 6th Successor (North-West)

            if (isValid(i - 1, j - 1)) {

                if (isDestination(i - 1, j - 1, dest)) {

                    cellDetails[i - 1][j - 1].parent\_i = i;

                    cellDetails[i - 1][j - 1].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i - 1][j - 1] && isUnBlocked(grid, i - 1, j - 1)) {

                    gNew = cellDetails[i][j].g + 1.414;

                    hNew = calculateHValue(i - 1, j - 1, dest);

                    fNew = gNew + hNew;

                    if                    (cellDetails[i - 1][j - 1].f == Double.POSITIVE\_INFINITY || cellDetails[i - 1][j - 1].f > fNew) {

                        openList.put(fNew, new int[]{i - 1, j - 1});

                        cellDetails[i - 1][j - 1].f = fNew;

                        cellDetails[i - 1][j - 1].g = gNew;

                        cellDetails[i - 1][j - 1].h = hNew;

                        cellDetails[i - 1][j - 1].parent\_i = i;

                        cellDetails[i - 1][j - 1].parent\_j = j;

                    }

                }

            }

            // 7th Successor (South-East)

            if (isValid(i + 1, j + 1)) {

                if (isDestination(i + 1, j + 1, dest)) {

                    cellDetails[i + 1][j + 1].parent\_i = i;

                    cellDetails[i + 1][j + 1].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i + 1][j + 1] && isUnBlocked(grid, i + 1, j + 1)) {

                    gNew = cellDetails[i][j].g + 1.414;

                    hNew = calculateHValue(i + 1, j + 1, dest);

                    fNew = gNew + hNew;

                    if (cellDetails[i + 1][j + 1].f == Double.POSITIVE\_INFINITY || cellDetails[i + 1][j + 1].f > fNew)

                        openList.put(fNew, new int[]{i + 1, j + 1});

                        cellDetails[i + 1][j + 1].f = fNew;

                        cellDetails[i + 1][j + 1].g = gNew;

                        cellDetails[i + 1][j + 1].h = hNew;

                        cellDetails[i + 1][j + 1].parent\_i = i;

                        cellDetails[i + 1][j + 1].parent\_j = j;

                    }

                }

            }

            // 8th Successor (South-West)

            if (isValid(i + 1, j - 1)) {

                if (isDestination(i + 1, j - 1, dest)) {

                    cellDetails[i + 1][j - 1].parent\_i = i;

                    cellDetails[i + 1][j - 1].parent\_j = j;

                    System.out.println("The destination cell is found");

                    tracePath(cellDetails, dest);

                    foundDest = true;

                    return;

                } else if (!closedList[i + 1][j - 1] && isUnBlocked(grid, i + 1, j - 1)) {

                    gNew = cellDetails[i][j].g + 1.414;

                    hNew = calculateHValue(i + 1, j - 1, dest);

                    fNew = gNew + hNew;

                    if (cellDetails[i + 1][j - 1].f == Double.POSITIVE\_INFINITY || cellDetails[i + 1][j - 1].f > fNew) {

                        openList.put(fNew, new int[]{i + 1, j - 1});

                        cellDetails[i + 1][j - 1].f = fNew;

                        cellDetails[i + 1][j - 1].g = gNew;

                        cellDetails[i + 1][j - 1].h = hNew;

                        cellDetails[i + 1][j - 1].parent\_i = i;

                        cellDetails[i + 1][j - 1].parent\_j = j;

                    }

                }

            }

        }

        if (!foundDest)

            System.out.println("Failed to find the destination cell");

    }

}

