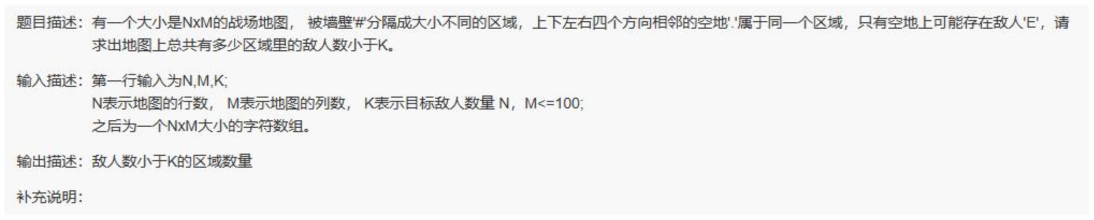
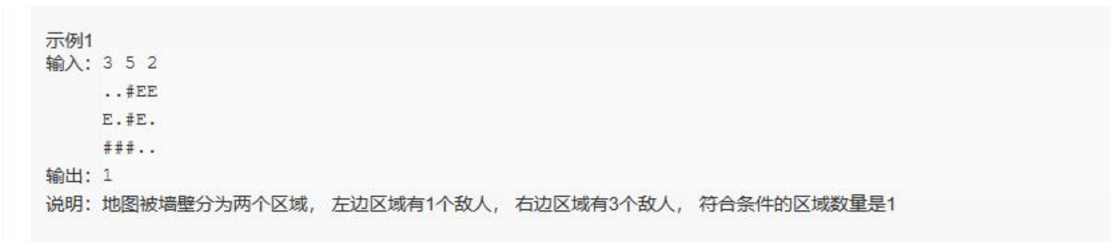
战 场 索 敌


题目描述：有一个大小是NxM的战场地图，被墙壁'#'分隔成大小不同的区域，上下左右四个方向相邻的空地'.'属于同一个区域，只有空地上可能存在敌人'E'，请求出地图上总共有多少区域里的敌人数小于K。

输入描述：第一行输入为N,M,K;
N表示地图的行数，M表示地图的列数，K表示目标敌人数量 N，M<=100;
之后为一个NxM大小的字符数组。

输出描述：敌人数小于K的区域数量

补充说明：


示例1
输入：3 5 2
```
..#EE
E.#E.
###..
```
输出：1
说明：地图被墙壁分为两个区域，左边区域有1个敌人，右边区域有3个敌人，符合条件的区域数量是1

```
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
        public static int row;
        public static int column;
        public static int enemy;
        public static boolean[][] alreadySearch;
        public static int[][] walkPosition = new int[][] {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};

        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                row = sc.nextInt();
                column = sc.nextInt();
                enemy = sc.nextInt();
                sc.nextLine();
                char[][] battleMap = new char[row][column];
                for (int i = 0; i < row; i++) {
                        battleMap[i] = sc.nextLine().toCharArray();
                }
                alreadySearch = new boolean[row][column];
                int enemyInArea = 0;
                for (int i = 0; i < row; i++) {
                        for (int j = 0; j < column; j++) {
                                if (alreadySearch[i][j] || battleMap[i][j] == '#') {
                                        continue;
                                }
                                if (searchEnemy(i, j, battleMap, row, column) < enemy) {
                                        enemyInArea++;
                                }
                        }
```

```java
        }
        System.out.println(enemyInArea);
    }

    private static int searchEnemy(int i, int j, char[][] battleMap, int row,
                                                                int column) {
        int enemyCount = 0;
        alreadySearch[i][j] = true;
        if (battleMap[i][j] == 'E') {
            enemyCount++;
        }
        LinkedList<int[]> positionStack = new LinkedList<>();
        positionStack.add(new int[] {i, j});
        while (positionStack.size() > 0) {
            int[] position = positionStack.removeLast();
            int x = position[0];
            int y = position[1];
            for (int k = 0; k < 4; k++) {
                int nextX = x + walkPosition[k][0];
                int nextY = y + walkPosition[k][1];
                if (nextX >= 0 && nextX < row
                        && nextY >= 0 && nextY < column
                        && !alreadySearch[nextX][nextY]
                        && battleMap[nextX][nextY] != '#') {
                    alreadySearch[nextX][nextY] = true;
                    if (battleMap[nextX][nextY] == 'E') {
                        enemyCount++;
                    }
                    positionStack.add(new int[] {nextX, nextY});
                }
            }
        }
        return enemyCount;
    }
}
```