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

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

```
#include <iostream>

#include <vector>

#include <string>
```

```
using namespace std;


vector<int> split(string str){
    vector<int> v;
    while(str.find(" ") != string::npos){
        int found = str.find(" ");
        v.push_back(stoi(str.substr(0, found)));
        str = str.substr(found + 1);
    }
    v.push_back(stoi(str));
    return v;
}


int dfs(vector<string>& board, vector<vector<bool>>& visited, int row, int col){
    int left, right, up, down, self = 0;
    left = right = up = down = 0;
    if(board[row][col] == 'E'){
        self = 1;
    }
    if(row > 0 && board[row - 1][col] != '#' && !visited[row - 1][col]){
        visited[row - 1][col] = true;
        up = dfs(board, visited, row - 1, col);
```

```
    }
    if(row < visited.size() - 1 && board[row + 1][col] != '#' && !visited[row + 1][col]){
        visited[row + 1][col] = true;
        down = dfs(board, visited, row + 1, col);
    }
    if(col > 0 && board[row][col - 1] != '#' && !visited[row][col - 1]){
        visited[row][col - 1] = true;
        left = dfs(board, visited, row, col - 1);
    }
    if(col < visited[0].size() - 1 && board[row][col + 1] != '#' && !visited[row][col + 1]){
        visited[row][col + 1] = true;
        right = dfs(board, visited, row, col + 1);
    }
    return self + left + right + up + down;
}

int main() {
    int N, M, K, temp, count = 0;
    string input_str;
    getline(cin, input_str);
```

```
vector<int> v = split(input_str);
N = v[0];
M = v[1];
K = v[2];
vector<string> board;
for(int i = 0; i < N; i++){
    getline(cin, input_str);
    board.push_back(input_str);
}
vector<vector<bool>> visited(N, vector<bool>(M, false));
for(int i = 0; i < N; i++){
    for(int j = 0; j < M; j++){
        if(board[i][j] != '#' && !visited[i][j]){
            visited[i][j] = true;
            temp = dfs(board, visited, i, j);
            if(temp < K){
                count++;
            }
        }
    }
}
cout << count;
```

```
    return 0;
}
// 64 位输出请用 printf("%lld")
```