1248：Dungeon Master

题目描述

这题是一个三维的迷宫题目，其中用‘.’表示空地，‘#’表示障碍物，‘S’表示起点，‘E’表示终点，求从起点到终点的最小移动次数，解法和二维的类似，只是在行动时除了东南西北移动外还多了上下。可以上下左右前后移动，每次都只能移到相邻的空位，每次需要花费一分钟，求从起点到终点最少要多久。

输入格式

多组测试数据。

一组测试测试数据表示一个三维迷宫：

前三个数，分别表示层数、一个面的长和宽，后面是每层的平面图。前三个数据为三个零表示结束。

输出格式

最小移动次数。

输入样例

```
3 4 5
S....
.###.
.##..
###.#
#####
#####
##.##
##...
#####
#####
#.###
####E
1 3 3
S##
#E#
###
0 0 0
```

输出样例

```
Escaped in 11 minute(s).
Trapped!
```

解析

本题的求解目标是最小移动步数，典型的广搜题目，需要注意的是，本题在原本的四方向的基础上还增加了两个维度，现在总共是上、下、左、右、前、后六个方向。

编码

```
#include<bits/stdc++.h>

using namespace std;

struct node {
    int z;
    int x;
    int y;
    int step;

    node() {}

    node(int z1, int x1, int y1, int step1) :
    z(z1), x(x1), y(y1), step(step1) {}
};

int l, n, m;
int Vis[35][35][35];
char Maps[35][35][35];
//定义六个方向
int u[6][3] = {{1,  0,  0}, //上
               {-1, 0,  0}, //下
               {0,  1,  0}, //右
               {0,  -1, 0}, //左
               {0,  0,  1}, //前
               {0,  0,  -1}}; //后

void bfs(node s, node e) {
    //true代表没有找到终止节点
    bool flag = true;
    //待查询队列表
    queue<node> Q;
    //将起始节点存入队列
    Q.push(node(s.z, s.x, s.y, s.step));
    //队列内存在待判断的点
    while (!Q.empty()) {
        node a = Q.front();
        Q.pop();
        //向六个方向查询
       for (int i = 0; i < 6; i++) {
            int zz = a.z + u[i][0];
```

```
            int xx = a.x + u[i][1];
            int yy = a.y + u[i][2];
            //三个条件：
          //1、没有越界
          //2、未被访问
          //3、可以通行
          if (zz >= 0 && zz < l && xx >= 0 && xx < n
                && yy >= 0 && yy < m && (!Vis[zz][xx][yy]) &&
                Maps[zz][xx][yy] != '#') {
                if (zz == e.z && xx == e.x && yy == e.y) {
                    printf("Escaped in %d minute(s).\n", a.step + 1)
                    flag = false;
                    return;
                }
                //将新的节点加入待搜索队列
             Q.push(node(zz, xx, yy, (a.step + 1)));
                Vis[zz][xx][yy] = 1;

            }
        }
    }
    //没有找到终点，无法通行
   if (flag) {
        printf("Trapped!\n");
    }
}

int main() {
    while (scanf("%d %d %d", &l, &n, &m)) {
        if (l + n + m == 0)
            break;
        node s, e;
        //数据的初始化
       memset(Vis, 0, sizeof(Vis));
        for (int k = 0; k < l; k++) {
            for (int i = 0; i < n; i++) {
                scanf("%s", &Maps[k][i]);
            }
        }
        for (int k = 0; k < l; k++) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < m; j++) {
                    if (Maps[k][i][j] == 'S') {
                        s.z = k;
                        s.x = i;
                        s.y = j;
                        s.step = 0;
                    }
                    if (Maps[k][i][j] == 'E') {
```

```
                        e.z = k;
                        e.x = i;
                        e.y = j;
                    }
                }
            }
        }
        //调用广度优先搜索
       bfs(s, e);
    }

    return 0;
}
```