1217：棋盘问题

题目描述

在一个给定形状的棋盘（形状可能是不规则的）上面摆放棋子，棋子没有区别。要求摆放时任意的两个棋子不能放在棋盘中的同一行或者同一列，请编程求解对于给定形状和大小的棋盘，摆放 k 个棋子的所有可行的摆放方案 C。

输入格式

输入含有多组测试数据。

每组数据的第一行是两个正整数n,k，用一个空格隔开，表示了将在一个n×n的矩阵内描述棋盘，以及摆放棋子的数目。 (n≤8,k≤n)

当为-1-1时表示输入结束。

随后的n行描述了棋盘的形状：每行有n个字符，其中 # 表示棋盘区域，. 表示空白区域（数据保证不出现多余的空白行或者空白列）。

输出格式

对于每一组数据，给出一行输出，输出摆放的方案数目C（数据保证C<2^{31}）。

输入样例

```
2 1
#.
.#
4 4
...#
..#.
.#..
#...
-1 -1
```

输出样例

```
2
1
```

解析

八皇后问题的变形，对于摆放的限定有：不能同行，不能同列，只能在"#"号中进行摆放。

编码

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

#define N 10
using namespace std;
int n, k; //棋盘的大小和需要摆放的棋子数
char maps[N][N]; //棋盘
int vis[N]; //访问列表
int cnt;

//x当前摆放的行
//y是当前摆放的棋子数量
void dfs(int x, int y) {
    //摆放的棋子数量等于目标值，则记录一种方法
    if (y == k) {
        cnt++;
        return;
    }
    //不断的尝试行列
    for (int i = x; i <= n; i++)
        for (int j = 1; j <= n; j++)
            //检测边界条件
            if (maps[i][j] == '#' && vis[j]) {
                //更新地图状态
                vis[j] = 0;
                //向下搜索
                dfs(i + 1, y + 1);
                //还原
                vis[j] = 1;
            }
    return;
}

int main() {
    //循环输入
    while (scanf("%d%d", &n, &k) != EOF) {
        if (n == -1 && k == -1)
            break;
        //多组测试数据，不要忘记重置
        memset(vis, 1, sizeof(vis));
        //读入地图数据
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                cin >> maps[i][j];
            }
        }
        //重置当前数据下的方案总数
        cnt = 0;
```

```
        dfs(1, 0);
        cout << cnt << endl;
    }
    return 0;
}
```