解题报告

参考答案

```
#include<iostream>
#define N 10
using namespace std;
int dx[4]={0,1,0,-1};
int dy[4]={1,0,-1,0};
char ch,d[N][N];//ch 记录用户点的符号
int a,b,n=5;
void dfs(int x,int y){
    d[x][y]='G';
    for(int k=0;k<4;k++){
        int nx=x+dx[k],ny=y+dy[k];
        if(nx>=1&&nx<=n&&ny>=1&&ny<=n&&d[nx][ny]==ch&&d[nx][ny]!='G')
            dfs(nx,ny);
    }
}
int main(){
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            cin>>d[i][j];
    cin>>a>>b;
    ch=d[a][b];
    dfs(a,b);
    for(int i=1;i<=n;i++,cout<<endl)
        for(int j=1;j<=n;j++)cout<<d[i][j];
    return 0;
}
```

题目分析

题目考察四方向连通 DFS 的理解和运用，需要注意：

1. dfs 递归函数一开始就要对地图当前位置赋值为 G

2. dfs 函数发生递归调用去搜索新位置的条件是：新位置不是 G。也即已经是金子的的地方不需要再进行搜索了，以免发生无限递归。

提示

题目是课件例子“发洪水”的拓展，地图上有多种符号，每一种符号都可以看成“空地”，由点下去的第一点(x,y)是哪种符号来决定，其余种类的符号则可以都看成“墙体”。

仔细观察第二个样例输入，会发现地图中初始就可能存在 G，写程序时要考虑到这种情况。

可参考课件第 7 页和第 10 页的程序。

易错点

1）忘记在 dfs 函数一开始对地图当前位置赋值 G。

2）没考虑地图中初始就存在 G，第一个点的位置符号 ch 正好是 G 的情况，递归调用去搜索新位置的条件漏掉了 `d[nx][ny]!='G'`，结果造成了无限递归。

解题报告

参考答案

```
#include<iostream>
#define N 110
using namespace std;
int dx[8]={-1,-1,-1,0,0,1,1,1};
int dy[8]={-1,0,1,-1,1,-1,0,1};
char d[N][N];
int n,m;
void dfs(int x,int y){
    d[x][y]='.';
    for(int k=0;k<8;k++){
        int nx=x+dx[k],ny=y+dy[k];
        if(nx>=1&&nx<=n&&ny>=1&&ny<=m&&d[nx][ny]=='@')
            dfs(nx,ny);
    }
}
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>d[i][j];
    int ans=0;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            if(d[i][j]=='@') {
                ans++;
                dfs(i,j);
            }
    cout<<ans<<endl;
    return 0;
}
```

题目分析

题目考察八方向连通区域计数的理解和运用，需要注意：

1. dfs 递归函数一开始，就要把地图当前位置表示有水的@变为表示没有水的小数点，也即标记这个位置已经搜索过了。

2. 主程序逐行逐列检查地图每个位置，一旦发现@，水坑数目加 1，随即展开 dfs 搜索附近八连通的@，把水坑连通区域的水都变成没水。

提示

题目是课件例子“数连通区域个数”的拓展。比较简单的方法是：扫描地图，每次发现一个新水坑的第一个@，计数器加 1，然后一边 dfs 搜索连通的@一边修改符号为小数点。

可参考课件第 8 页和第 23 页的程序。

本题也可以使用 vst 数组来记录已经搜索过的位置。

易错点

1）忘记在 dfs 函数一开始把地图当前位置的水“抽干”。
2）没有标记已经搜索过的位置，造成重复计数。

解题报告

参考答案

```
#include<iostream>
#define N 20
using namespace std;
int dx[4]={0,1,0,-1};
int dy[4]={1,0,-1,0};
char d[N][N];
int area,n=10,m=10;
void dfs(int x,int y){
    if(x<1||x>n||y<1||y>m)return;
    d[x][y]='.';
    area++;
    for(int k=0;k<4;k++){
        int nx=x+dx[k],ny=y+dy[k];
        if(d[nx][ny]=='#') dfs(nx,ny);//有越界风险
    }
}
int main(){
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>d[i][j];
    int ans=0;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            if(d[i][j]=='#'){
                area=0;
                dfs(i,j);
                ans+=area>1;
            }
    cout<<ans<<endl;
    return 0;
}
```

题目分析

题目考察累计连通区域面积的理解和运用，需要注意：

1. dfs 递归函数需要注意当前搜索位置(x,y)不能越界。
2. dfs 搜索时通过修改符号来标记已搜索过的位置，也可以使用 vst 数组记录搜索信息。
3. 连通区域的面积必须大于 1，才认为是有效区域，计数加 1。

提示

题目是课件例子“累计连通区域面积”的拓展。主程序逐行逐列检查地图每个位置，一旦发现#，随即展开 dfs 搜索附近四连通区域，并累计其面积。如果面积为 1，则认为是灰尘。如果超过 1，则计数器加 1。搜索过的#全部改为.。

可参考课件第 22 页的程序。

易错点

1） 搜索位置越界
2） 没有标记已经搜索过的位置，造成重复计数。
3） 把面积为 1 的灰尘当成有效区域，计数过多。