解题报告

参考答案

```
#include<iostream>
#include<algorithm>
#define N 8
using namespace std;
int ans;
bool clmn[N],row[N],d1[2*N],d2[2*N];
void dfs(int x){
    if(x==N){ans++;return;}
    if(row[x]){dfs(x+1);return;}
    for(int y=0;y<N;y++)
        if(!clmn[y]&&!d1[x+y]&&!d2[x-y+N-1]){
            clmn[y]=d1[x+y]=d2[x-y+N-1]=1;
            dfs(x+1);
            clmn[y]=d1[x+y]=d2[x-y+N-1]=0;
        }
}
int main(){
    ans=0;
    for(int i=0;i<N;i++)
        for(int j=0;j<N;j++){
            char ch;
            cin>>ch;
            if(ch=='Q')row[i]=clmn[j]=d1[i+j]=d2[i-j+N-1]=1;
        }
    dfs(0);
    cout<<ans<<endl;
    return 0;
}
```

题目分析

题目考察八皇后问题的理解和运用，需要注意棋盘上初始放置的皇后的影响，对课件例子程序进行相应修改。

提示

题目是课件例子“八皇后逐行放”的拓展，可参考课件第 19 页的程序。

利用课件已提供的程序，进行程序填空。根据棋盘初始已放置的皇后，修改 row、clmn、d1、d2 数组的相应元素，进行标记。dfs 函数中需要考虑如果某行初始已有皇后，则跳过该行，转入下一行的搜索。

易错点

1）读入棋盘上皇后时，忘记了对 row、clmn、d1、d2 数组的相应元素进行修改。

2）若第 x 行初始已有皇后，直接转入第 x+1 行进行搜索，然后返回。忘记了返回会造成重复搜索。

解题报告

参考答案

```
#include<bits/stdc++.h>
#define N 10
using namespace std;
char d[N][N];
int n,k,ans=0,clmn[N];
void dfs(int x,int num){//从第 x 行开始放，已经放了 num 个
    if(num==k) {ans++;return;}
    if(x==n)return;
    for(int y=0;y<n;y++)
        if(!clmn[y]&&d[x][y]=='o'){
            clmn[y]=1;
            dfs(x+1,num+1);
            clmn[y]=0;
        }
    dfs(x+1,num);//第 x 行不放了，从第 x+1 行开始放
}
int main(){
    cin>>n>>k;
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            cin>>d[i][j];
    dfs(0,0);
    cout<<ans<<endl;
    return 0;
}
```

题目分析

题目考察八皇后问题的理解和运用，需要注意：

1. 比八皇后简单的地方在于：不用考虑两条斜线方向。
2. 比八皇后稍微复杂的地方在于：只有棋盘上是 o 的地方可以摆放棋子。

提示

参考课件例子“八皇后逐行放”，按行摆放棋子。利用课件已提供的程序，进行程序填空。dfs(x,num)表示已经摆放了 num 枚棋子，当前考察第 x 行。如果 num 等于 k，则 k 枚棋子摆放完毕，答案数加 1，函数结束。如果 x 等于 n，则表示最后一行已经考察过了，现在要越界了，函数结束。然后，考察第 x 行的每一列，如果某一列 y 没有同列棋子，并且第 x 行 y 列位置上是 o，则该位置可以放棋子。标记该列为已有棋子，递归调用 dfs(x+1,num+1)。

易错点

1）棋盘上是 o 的地方，方可摆放棋子。

2）搜索完第 x 行的每一列后，还要递归调用 dfs(x+1,num)。体现第 x 行没有放棋子的情形，直接转入考察第 x+1 行，已放置棋子数依然是 num。

解题报告

参考答案

```
#include<iostream>
#include<algorithm>
#define N 10
using namespace std;
int ans=0,n=5,f[N][N];
char ditu[N][N];
bool valid(int x,int y){
    for(int i=x-1;i>=0;i--)
        if(ditu[i][y]=='X')break;
        else if(f[i][y])return 0;
    for(int j=y-1;j>=0;j--)
        if(ditu[x][j]=='X')break;
        else if(f[x][j])return 0;
    return 1;
}
void dfs(int x,int y,int c){
    ans=max(ans,c);
    if(x==n)return;
    int nx=(y==n-1?x+1:x);
    int ny=(y==n-1?0:y+1);
    if(ditu[x][y]=='.'&&valid(x,y)){
        f[x][y]=1;
        dfs(nx,ny,c+1);
        f[x][y]=0;
    }
    dfs(nx,ny,c);
}
int main(){
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            cin>>ditu[i][j];
    dfs(0,0,0);
    cout<<ans<<endl;
    return 0;
}
```

题目分析

题目考察八皇后问题的理解和运用，需要注意：

1. 比八皇后简单的地方在于：不用考虑两条斜线方向。
2. 比八皇后复杂的地方在于：棋盘上的 x 表示墙，可以挡住子弹，从而使墙的两侧可以各放置一个城堡。

提示

参考课件例子“八皇后逐行放”，按行摆放城堡。利用课件已提供的程序，进行程序填空。

定义 valid 函数，用于判断 x 行 y 列是否可以放置城堡：由近及远依次检查第 y 列的上方格子，如果先遇到墙，则返回 true；如果先遇到城堡，则返回 false。同理，由近及远依次检查第 x 行的左方格子。

dfs(x,y,c)表示已经摆放了 c 座城堡，当前考察第 x 行第 y 列这个格子：用 c 去更新当前最大城堡数（如果 c 更大的话）。如果 x 等于 n，则表示最后一行已经考察过了，函数结束。然后，计算出下一位置(nx,ny)。如果当前位置(x,y)是空地且可以放置城堡，则标记当前位置有城堡，递归调用 `dfs(nx,ny,c+1)`。

易错点

1）判断某位置是否 valid，只要向上向左搜索其他城堡，若先遇到墙，则不再搜索。

2) dfs 函数最后需要考虑当前位置(x,y)不可放置城堡的情形，要递归调用 `dfs(nx,ny,c)`。