AcWing-4737：冰壶

红队和黄队进行了一场冰壶比赛，比赛结束后，裁判正在计算两队的得分。

场地可以看作一个二维平面，得分区域可以看作一个以
$(0, 0)$ 为圆心
R_h 为半径的圆。

场地上散落着
N 个红队的冰壶以及
M 个黄队的冰壶。

冰壶可以看作一个半径为 R_s 的圆。

每个冰壶的圆心坐标已知。

如果一个冰壶的任何部分位于得分区域的圆上或圆内（两者相切也算），则视为该冰壶位于得分区域内。

如果一个冰壶能够同时满足：

1. 它位于得分区域内。
2. 不存在任何**对方**冰壶比它距离得分中心 $(0, 0)$ 更近（欧几里得距离）。

那么，这个冰壶就是一个得分冰壶。

一个队伍的最终得分等于该队伍的得分冰壶数量。

请你计算并输出两支队伍的最终得分。

一个冰壶与得分中心之间的距离等于其圆心点到点 $(0, 0)$ 的距离。

数据保证不同冰壶与得分中心之间的距离不同，且冰壶两两之间不重叠（但可能相切）。

输入格式

第一行包含整数
T，表示共有
T 组测试数据。

每组数据第一行包含两个整数 R_s, R_h。

接下来一行包含一个整数 N。

接下来
N 行，每行包含两个整数
X_i, Y_i，表示一个红队冰壶的圆心坐标为
(X_i, Y_i)。

接下来一行包含一个整数 M。

接下来
M 行，每行包含两个整数
Z_i, W_i，表示一个黄队冰壶的圆心坐标为
(Z_i, W_i)。

输出格式

每组数据输出一个结果，每个结果占一行。

结果表示为 `Case #x: y z`，其中
x 为组别编号（从
1 开始），
y 为红队得分，
z 为黄队得分。

数据范围

$1 \le T \le 100$,
$1 \le R_s < R_h \le 10^4$,
$0 \le N, M \le 8$,
$-20000 \le X_i, Y_i, Z_i, W_i \le 20000$

时/空限制：	1s / 64MB
总通过数：	776
总尝试数：	1648
来源：	Google Kickstart2022 Round G Problem B
算法标签	

输入样例1：

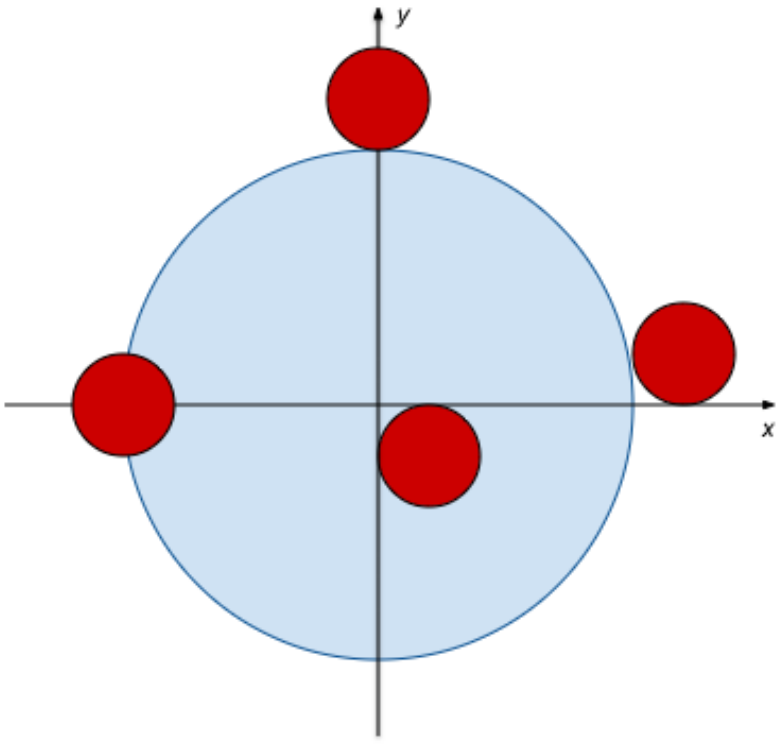
```
2
1 5
4
1 -1
6 1
0 6
-5 0
0
10 100
2
-3 -4
200 200
0
```

输出样例1：

```
Case #1: 3 0
Case #2: 1 0
```

样例1解释

Case 1 的冰壶分布情况如下图所示。



在这种情况下，黄队没有冰壶在得分区域内，所以红队在得分区域内的每个冰壶都是得分冰壶。

红队除了以 (6,1) 为圆心的那个冰壶以外，其他所有的冰壶都在得分区域内，所以红队得 3 分。

输入样例2：

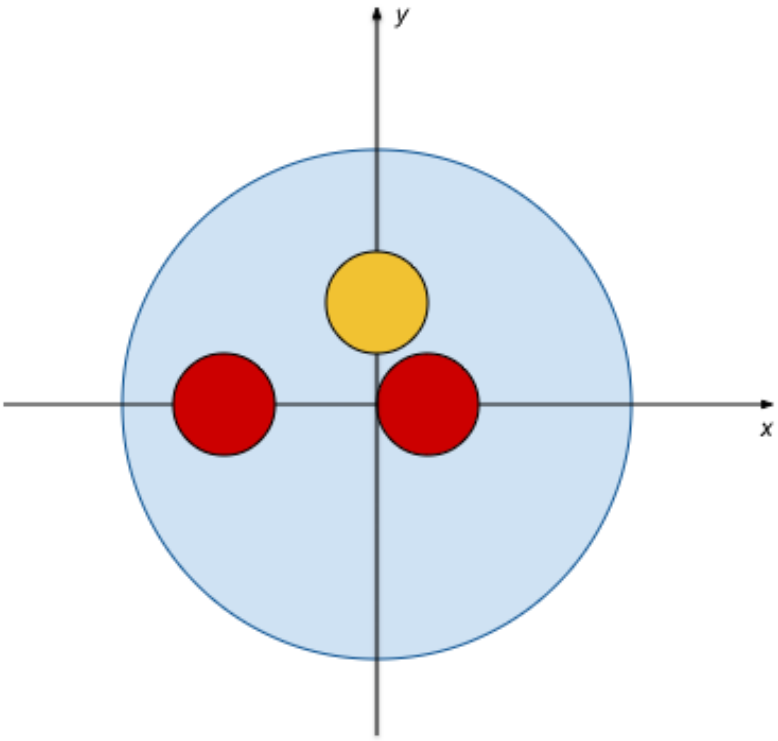
```
2
1 5
2
1 0
-3 0
1
0 2
10 50
2
-40 -31
-35 70
3
59 0
-10 0
30 40
```

输出样例2：

```
Case #1: 1 0
Case #2: 0 2
```

样例2解释

Case 1 的冰壶分布情况如下图所示。



在这种情况下，两支队伍都有冰壶在得分区域内。

红队圆心为 (1,0) 的冰壶在得分区域内，且没有黄队的冰壶比它距离得分中心更近，所以它值 1 分。

红队圆心为 (−3,0) 的冰壶在得分区域内，但是黄队圆心为 (0,2) 的冰壶比它距离得分中心更近，所以它不能得分。

黄队圆心为 (0,2) 的冰壶在得分区域内，但是红队圆心为 (1,0) 的冰壶比它距离得分中心更近，所以它不能得分。

因此，红队得 1 分，黄队得 0 分。

解题思路：非常简单的模拟+枚举问题，主要思路为分别存储红队和黄队**在得分区内**的冰壶位置，使用`vector<pair<int,int>>`存储，并在输入时分别记录两队得分区内距离原点最近的冰壶。

当输入和存储完毕，只需要比较两队距离原点最近的冰壶分别的距离，如果不一致，那么距离大的必定得分为0，接着计算距离小的一队得分即可。如果距离相等，那么会有两种情况——都不在得分区，或都有得分区内的冰壶且最小距离相等，那么分别计算即可。

Code：

```
//
//  main.cpp
//  4737-冰壶
//
//  Created by MacBook Pro on 2023/7/23.
//

#include <iostream>
#include <vector>
#include <climits>
using namespace std;

int T;

vector<pair<int,int>> red;
vector<pair<int,int>> yel;


int main() {
    scanf("%d",&T);
    for(int cases=1;cases<=T;cases++){
        red.clear(); yel.clear();
        int rs,rh,n,m;
        int x1=-1,y1=-1,x2=-1,y2=-1;
        int len1=INT_MAX,len2=INT_MAX;
        scanf("%d%d",&rs,&rh);
        //red
        scanf("%d",&n);
        for(int i=0;i<n;i++){
            int x,y;
            scanf("%d%d",&x,&y);
            if(x*x+y*y<=(rs+rh)*(rs+rh)){
                //得分区内
```

```
                red.push_back({x,y});
                if(x*x+y*y<len1){
                    len1=x*x+y*y;
                    x1=x; y1=y;
                }
            }
        }
        //yellow
        scanf("%d",&m);
        for(int i=0;i<m;i++){
            int x,y;
            scanf("%d%d",&x,&y);
            if(x*x+y*y<=(rs+rh)*(rs+rh)){
                //得分区内
                yel.push_back({x,y});
                if(x*x+y*y<len2){
                    len2=x*x+y*y;
                    x2=x; y2=y;
                }
            }
        }
        //conclude
        if(len1<len2){
            //red更有优势
            int ans=0;
            for(int i=0;i<red.size();i++){
                int x=red[i].first,y=red[i].second;
                if(x*x+y*y<=len2) ans++;
            }
            printf("Case #%d: %d 0\n",cases,ans);
        }
        else if(len1>len2){
            //yellow更有优势
            int ans=0;
            for(int i=0;i<yel.size();i++){
                int x=yel[i].first,y=yel[i].second;
                if(x*x+y*y<=len1) ans++;
            }
            printf("Case #%d: 0 %d\n",cases,ans);
        }
        else{
            //相同
            int ans1=0,ans2=0;
            for(int i=0;i<yel.size();i++){
                int x=yel[i].first,y=yel[i].second;
                if(x*x+y*y<=len1) ans1++;
            }
            for(int i=0;i<red.size();i++){
                int x=red[i].first,y=red[i].second;
```

```
                if(x*x+y*y<=len2) ans2++;
            }
            printf("Case #%d: %d %d\n",cases,ans1,ans2);
        }
    }
    return 0;
}
```

代码提交状态：Accepted