C6 - Solution

A - 正确率

难度	考点
2	条件，字符串

题目分析

每读入一条信息，统计其是否是AC，以及是否是CE。

分别记录AC和非CE的个数（只需判断字符串第一个字母即可），然后两者之比即为答案。

示例代码

```
#include<stdio.h>
int main()
{
    char s[10];
    int ac = 0,not_ce = 0;
    while(scanf("%s",s) != EOF)//读入一行不带空格字符串，以'\0'结尾
    {
        if(s[0] == 'A')//AC
            ac++;

        if(s[0] != 'C')//非CE
            not_ce++;
    }
    printf("%.3lf",1.0*ac/not_ce);//答案。注意1.0*是为了类型转换
    return 0;
}
```

B - Monica的策划

难度	考点
3	排序

题目分析

题意即要求把活跃度排序后输出p_i大。

参照ppt介绍的冒泡排序，把数组排序后输出答案即可。

示例代码

```
#include <stdio.h>
```

```
#include <string.h>
#define ll long long
#define MN (2000+5)
int a[MN];
int n,k;
int main(){
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;++i)
        scanf("%d",&a[i]);
    for(int i=1;i<=n;++i)                //冒泡排序
        for(int j=1;j<=n-i;++j)
            if(a[j]<a[j+1]){
                int tmp=a[j];
                a[j]=a[j+1];
                a[j+1]=tmp;
            }
    for(int i=1;i<=k;++i){
        int x;
        scanf("%d",&x);
        printf("%d\n",a[x]);
    }
    return 0;
}
```

C - sizeof

难度	考点
1	对数组长度sizeof的理解

题目分析

- 思路很简单，N 个基本类型，每个基本类型占字节数 b_i，数组长度 L，事实上 $N=1$ 相当于熟悉的数组求 `sizeof`。
- 直接模拟题目的 $S=\sum_{i=1}^{N} b_i$，求 $S\times L$ 输出即可。
- 注意判断输入什么类型时，当然可以用标准字符串库 `string.h` 里的 `strcmp() == 0` 来判断，注意到 `int`，`char`，`double`，`long long` 的首字母并不同，所以直接 `switch` 其首字母即可。
- 如果使用字符串比较请注意 `long long` 的空格，如果用 `scanf()` 可能会 WA 在这上面，请用 `gets()` 读取一整行。
- 一个生动有趣的讲解：

这是关于数组的一个案例教学：

现在有 5 位学生的成绩数值，要用 C 语言存下来。

华强说这简单，`int a[5] = {100,99,98,97,96};`，然后老师问这个数组占用了多少空间，宋老虎说这简单，`sizeof(a) = 20`，老师问为啥，老虎说，`int` 类型占 4 个字节，也就是 `sizeof(int)`，数组长度为 5，长度×`sizeof(类型)`，就乘出来了。

老师说好，然后提了个思考题，说这题具体 C 语言的实现还没有学，但是其思想可供参考，比如现在除了成绩还要记录其ABCDE的等级，怎么存。

振涛想了想虽然不知道怎么实现，但是他写了个 C 语言风格的数组：

```
{{100,'A'},{80,'B'},{70,'C'}}
```

而且他觉得这个数组如果符合 C 语言语法，用 `sizeof` 应该得的值是 15。

老师说其实思路差不多，你们的疑惑是不知道声明成什么类型，既不是int也不是char，虽然这一讲不会细说，但是可以理解成一个“组合类型”，它占的字节可以简单理解为组成该“组合类型”的基本类型占字节数的加和。

华强悟了，刚刚那个成绩与等级的组合，就是`int`（占 4 个字节）和 `char` 加起来 （占 1 个字节），这个“组合类型”理解为占 5 个字节，然后乘上长度 3 就是 15。

郝哥也提出了自己的见解，现在比如这样一个存储，同学的性别、学号和手机号，性别就是 `F` 和 `M` 一个 `char`，学号 8 位一个 `int`，手机号 11 位会溢出于是用一个 `long long`，加和一下这个“组合类型”占 1 + 4 + 8 = 11 个字节。数组长度为 5 的话 `sizeof` 返回的值就是 55。

示例代码

```c
#include<stdio.h>
int main() {
    char type[15];
    int n, len, size=0;
    scanf("%d%d\n", &n, &len);
    for(int i = 0; i < n; ++i){
        gets(type);
        switch(type[0]){
            case 'c': size+=1; break;
            case 'i': size+=4; break;
            case 'l': size+=8; break;
            case 'd': size+=8; break;
        }
    }
    printf("%d\n", len * size);
    return 0;
}
```

D - 字符串匹配

难度	考点
2	字符串

题目分析

由于字符串含空格，需要使用`gets()`函数来读取一行字符串。

之后枚举字符串 A 的每个位置，判断从该位置起的字符能否与字符串 B 一一对应，若可以，输出此时是 A 字符串的第几位，`return 0`即可。

示例代码

```
#include<stdio.h>
#include<string.h>

int lena,lenb;
char a[105],b[105];

int pd(int x){
    for(int i=0;i<lenb;i++)
        if(a[x+i]!=b[i])
            return 0;
    return 1;
}

int main(){
    gets(a),gets(b);
    lena=strlen(a),lenb=strlen(b);
    for(int i=0;i<=lena-lenb;i++){
        if(pd(i)){
            printf("YES %d",i);
            return 0;
        }
    }
    printf("NO");
    return 0;
}
```

E - 矩阵乘法

难度	考点
3	二维数组

题目分析

本题难度不大，只需按照题目要求进行模拟即可

一些注意事项已经写在了参考代码中。

示例代码

```
#include<stdio.h>
int a[21][21],b[21][21],c[21][21];
int n,t;

void multi()
{
    int sum=0;
```

```
    for(int i=1;i<=t;i++){
        for(int j=1;j<=t;j++){
            sum=0;
            for(int k=1;k<=t;k++)
                sum+=a[i][k]*b[k][j];
            c[i][j]=sum;
            //求a[i][j],注意求出的值不能直接赋给a[i][j]，
            //因为需要用到原先a[i][j]的值来求其它位置的值，如：a[i][j+1]
        }
    }

    for(int i=1;i<=t;i++)
        for(int j=1;j<=t;j++)
            a[i][j]=c[i][j];

    return;
}

int main()
{
    scanf("%d",&t);

    for(int i=1;i<=t;i++)//输入矩阵
        for (int j=1;j<=t;j++)
                scanf("%d",&a[i][j]);

    for(int i=1;i<=t;i++)//输入矩阵
        for (int j=1;j<=t;j++)
                scanf("%d",&b[i][j]);

    multi();//输入完后就进行矩阵乘法

    for(int i=1;i<=t;i++){//结果输出
        for(int j=1;j<=t;j++){
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

F - 应急食品

难度	考点
3	循环、二维数组

题目分析

本题中棋盘合法的条件是：

- 每行中都填有$1,2,\cdots,9$且不重复；
- 每列中都填有$1,2,\cdots,9$且不重复；
- 每一个3×3的九宫格中都填有$1,2,\cdots,9$且不重复。

对于不重复数字，我们可以开一个一维数组来记录每个数字在某一行或者某一列中出现了多少次，如果一个数字出现超过1次，那么说明棋盘不合法。

示例代码

```
#include <stdio.h>
const int f[3] = {0, 1, 2};
int flag = 1, num[10 + 5], map[10 + 5][10 + 5];
int main()
{
    int i, j, k, l, x, y, sum;
    for (i = 0; i < 9 && flag; i++)
    { //输入 + 判断每行是否合法
        for (j = 1; j <= 9; j++)
            num[j] = 0;
        for (j = 0; j < 9; j++)
        {
            scanf("%d", &map[i][j]);
            if (num[map[i][j]])
                flag = 0;
            else
                num[map[i][j]]++;
        }
    }
    for (i = 0; i < 9 && flag; i++)
    { //判断每列是否合法
        for (j = 1; j <= 9; j++)
            num[j] = 0;
        for (j = 0; j < 9 && flag; j++)
        {
            if (num[map[j][i]])
                flag = 0;
            else
                num[map[j][i]]++;
        }
    }
    for (i = 0; i < 9 && flag; i += 3)
        for (j = 0; j < 9 && flag; j += 3)
        { //判断每个九宫格是否合法
            for (k = 1; k <= 9; k++)
                num[k] = 0;
            for (k = 0; k < 3; k++)
                for (l = 0; l < 3; l++)
                {
                    x = i + f[k];
                    y = j + f[l];
                    if (num[map[x][y]])
                        flag = 0;
                    else
                        num[map[x][y]]++;
                }
        }
    printf(flag ? "Cheers! O(^_^)O\n" : "Emergency food! ~>_<~\n");
    return 0;
}
```

G - 二分查找·改

难度	考点
4	二分查找

题目分析

本题为二分查找的模版题。

定义当前被查找的区间为 $[L,R]$，$mid=\lfloor\frac{L+R}{2}\rfloor(mid<R)$，被查找数组为 $a[maxn]$，查找值为 e。

先讨论 `lower_bound`：

当 $a[mid]\geq e$ 时，答案一定在 $[L,mid]$ 区间内；当 $a[mid]<e$ 时，答案一定在 $[mid+1,R]$ 区间内。当$L==R$时停止查找。

再讨论 `upper_bound`：

当 $a[mid]>e$ 时，答案一定在 $[L,mid]$ 区间内；当 $a[mid]\leq e$ 时，答案一定在 $[mid+1,R]$ 区间内。当$L==R$时停止查找。

常见错误原因：

- 当 $a[mid]$ 与被查找值相等时使用类似 `while(a[mid] == e) mid--;` 的写法来寻找最小编号。在被查找数组大部分值相等时，上述做法的时间复杂度会退化成 $O(n)$，进而导致 TLE。
- 由于边界判断不当（例如将 $mid+1$ 错写为 mid）导致二分前和二分后得到区间相同，会导致递归无法停止/死循环。

示例代码

```
#include <stdio.h>
#define maxn 1000010
int a[maxn];
int lower_bound(int e, int l, int r)
{
    if (l == r)
    {
        if (a[l] < e)
            return -1;
        else
            return l;
    }
    int mid = (l + r) / 2;
    if (a[mid] >= e)
        return lower_bound(e, l, mid);
    else
        return lower_bound(e, mid + 1, r);
}
int upper_bound(int e, int l, int r)
{
    if (l == r)
    {
        if (a[l] <= e)
```

```
            return -1;
        else
            return l;
    }
    int mid = (l + r) / 2;
    if (a[mid] > e)
        return upper_bound(e, l, mid);
    else
        return upper_bound(e, mid + 1, r);
}
int main()
{
    int n, m, e;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);

    while (m--)
    {
        scanf("%d", &e);
        printf("%d ", lower_bound(e, 1, n));
        printf("%d\n", upper_bound(e, 1, n));
    }

    return 0;
}
```

H - 康威生命游戏

难度	考点
4	二维数组、模拟

题目分析

本题只需要照着题目里面说的步骤一步一步模拟即可。

在计算“周围有多少个活细胞”时需要注意一下目前正在计算的细胞是否超过了边界等不需要计算的情况。

在代码中，使用了$cnt[i,j]$记录了第i行第j列的细胞周围有多少个活细胞，然后直接更新棋盘$m[i,j]$即可。

示例代码

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include <stdlib.h>
#define ll long long
#define maxn 100+5
const double eps=1e-11;

int m[maxn][maxn];
int cnt[maxn][maxn];
```

```
int n,k;

int cn(int i,int j) {
    int ans=0;
    for(int si=i-1; si<=i+1; ++si) {
        for(int sj=j-1; sj<=j+1; ++sj) {
            if(si>=1 && si<=n && sj>=1 && sj<=n && !(si==i && sj==j))
                ans+=m[si][sj];
        }
    }
    return ans;
}

int main() {
    scanf("%d%d",&n,&k);
    for(int i=1; i<=n; ++i)
        for(int j=1; j<=n; ++j)
            scanf("%d",&m[i][j]);

    while(k--) {
        memset(cnt,0,sizeof(cnt));
        for(int i=1; i<=n; ++i) {
            for(int j=1; j<=n; ++j) {
                cnt[i][j]=cn(i,j);
            }
        }

        for(int i=1; i<=n; ++i) {
            for(int j=1; j<=n; ++j) {
                if(m[i][j]==0) {
                    if(cnt[i][j]==3)
                        m[i][j]=1;
                } else {
                    if(cnt[i][j]<2 || cnt[i][j]>3)
                        m[i][j]=0;
                }
            }
        }
    }
    for(int i=1;i<=n;++i){
        for(int j=1;j<=n;++j)
            printf("%d ",m[i][j]);
        printf("\n");
    }

    return 0;
}
```

I - 排序、贪心

难度	考点
5	排序、贪心

题目分析

写完作业的时间取决于所花时间**最长**的那项作业，因此本题是要找到一种作业的排列顺序，使得**最长时间最短。**

考察排列中任意两个相邻的作业：

作业	难度	作业量
$\cdots$	$\cdots$	$\cdots$
h_1	a_1	b_1
h_2	a_2	b_2
$\cdots$	$\cdots$	$\cdots$

可得 $ans_1 = max(\frac{m}{b_1}, \frac{m \times a_1}{b_2})$，其中 m 为精力 s 和前面所有作业的难度之积。

如果将这两作业调换顺序：

作业	难度	作业量
$\cdots$	$\cdots$	$\cdots$
h_2	a_2	b_2
h_1	a_1	b_1
$\cdots$	$\cdots$	$\cdots$

可得 $ans_2 = max(\frac{m}{b_2}, \frac{m \times a_2}{b_1})$

对比一下两个答案：

$ans_1 = max(\frac{m}{b_1}, \frac{m \times a_1}{b_2})$

$ans_2 = max(\frac{m}{b_2}, \frac{m \times a_2}{b_1})$

显然有 $\frac{m \times a_1}{b_2} \geq \frac{m}{b_2}, \frac{m \times a_2}{b_1} \geq \frac{m}{b_1}$

因此如果 $ans_1 \geq ans_2$，一定有 $\frac{m \times a_1}{b_2} \geq \frac{m \times a_2}{b_1}$

即 $a_1 \times b_1 \geq a_2 \times b_2$

为了 ans 取到较小值，需要将 $a_i \times b_i$ 较小的排在前面

所以以 $a_i \times b_i$ 为关键字对所有作业排序即可

下面再证明该排序方法对于整体也是最优解：

假设排列不是按 $a_i \times b_i$ 单调递增的，即存在 $i \in [1, s) \bigcap N$ 使得 $a_i \times b_i > a_{i+1} \times b_{i+1}$

根据上面的证明，将这两项交换所得到的答案一定小于或等于原先的答案

所以该方法的得到的一定是最优解

示例代码

```
#include <stdio.h>

int main()
{
    int n, s, tmp, a[1005], b[1005];
    scanf("%d %d", &n, &s);
    for (int i = 0; i < n; i ++)
        scanf("%d %d", &a[i], &b[i]);
    for (int i = n-1; i > 0; i --) //冒泡排序（或者用qsort）
        for (int j = 0; j < i; j ++)
            if (a[j]*b[j] > a[j+1]*b[j+1])
            {
                tmp = a[j], a[j] = a[j+1], a[j+1] = tmp;
                tmp = b[j], b[j] = b[j+1], b[j+1] = tmp;
            }
    long long mul = s, ans = 0;
    for (int i = 0; i < n; i ++) //寻找最长时间
    {
        if (mul / b[i] > ans)
            ans = mul / b[i];
        mul *= a[i];
    }
    printf("%lld", ans);
    return 0;
}
```

J - 圈Sheep

难度	考点
7	计算几何、数学计算

题目分析

本题的方法就是要对复杂的三角形与圆交点情况的分类，而分类的方式有很多种，在本题中已经通过输出提示了大家可以通过三角形顶点在圆内的个数和在圆外的个数分类，而核心便是求三角形在圆内的部分的面积

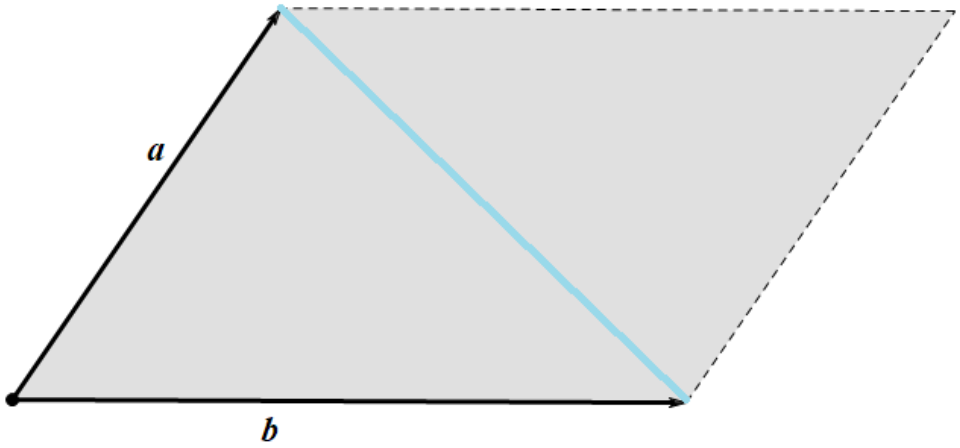
所以情况可以大致分4种

1. 三角形的所有顶点都在圆内，这个时候仅需要计算三角形面积，当然，这里可以使用大家在$E5$中使用的方法，但平面和三维三角形的面积更推荐大家使用向量外积计算

   在下图中$\vec{a} \times \vec{b}$即为由$\vec{a}, \vec{b}$向量构成的平行四边形的矢量面积，乘$\frac{1}{2}$并取绝对值即可得到三角形的面积

   $S_{\Delta} = \frac{1}{2}|\vec{a} \times \vec{b}|$

   详细可参考：【平面向量】向量的叉积与三角形的面积 - 知乎 (zhihu.com)



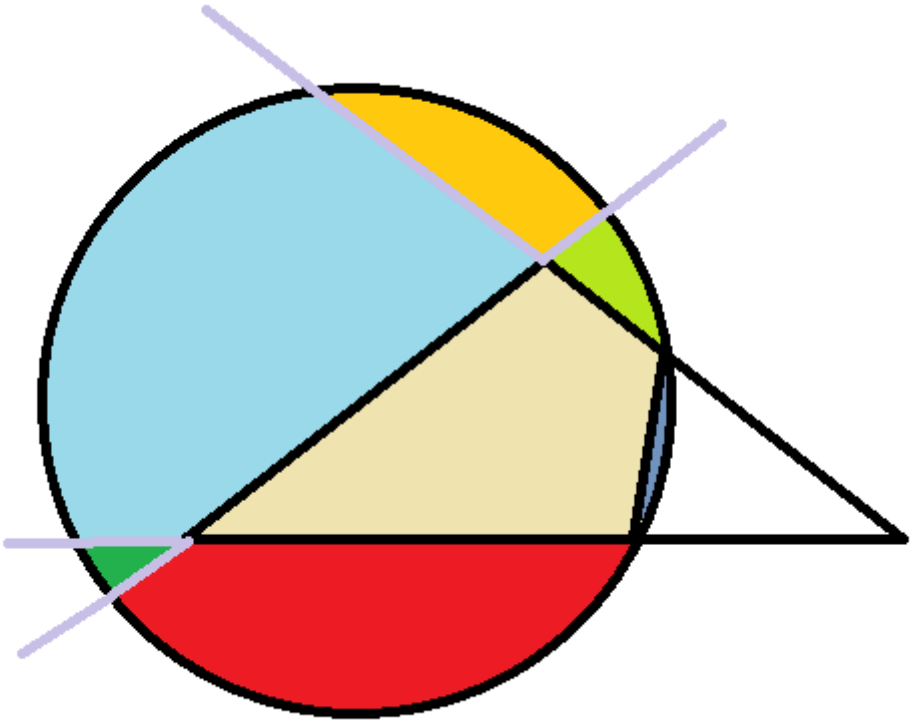
$$S = S_{\Delta}$$

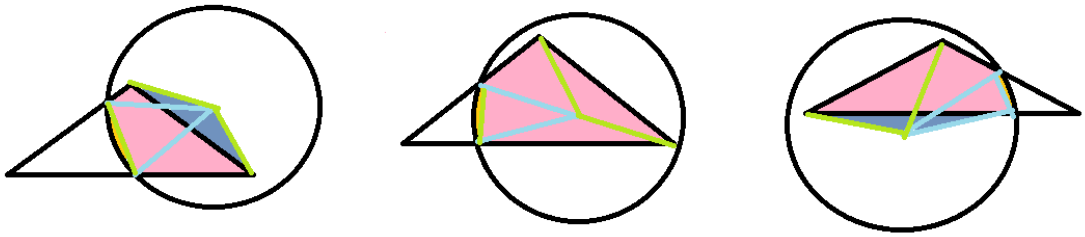
2. 三角形有两个顶点在圆内，这个时候可以大致将圆心的位置分配在不同的彩色区域进行计算，算上计算弧时的三角形，即为4个三角形的面积加减，这里仍推荐大家使用外积计算，利用不取模的外积运算我们可以直接通过外积产生的法向量的正负方向来简化部分情况的求解

$$S = S_{弧} + S_{\Delta 1} + S_{\Delta 2} + S_{\Delta 3} + S_{\Delta 4}$$

注意这里的面积因为是矢量乘法，所以有正有负。

我们可以通过直线把圆心分割出的各区域，大致分成如下的不同彩色区域所对应的情况



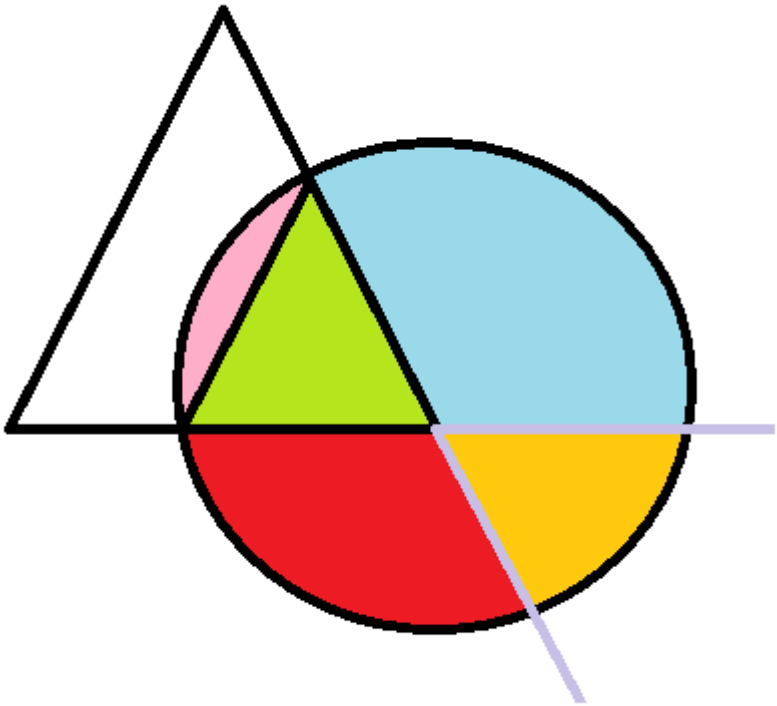


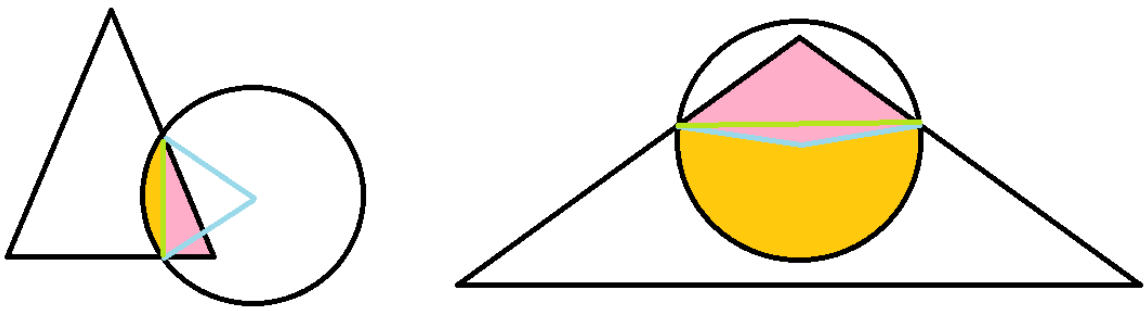
（上图中列出了几种（其他类似），所求面积为$S_{橙色}+S_{粉色}$，深蓝色部分即为需要减去的部分）

3. 三角形仅有一个顶点在圆内，这个时候，按照可以求解的方式，我们可以大致将所求解的面积分割为弦面积+与圆产生的交点与在圆内的顶点所形成的三角形的面积的和

$$S=S_{弧}+\sum S_{\Delta}$$

同样我们可以把圆心的位置分配在五个彩色区域，来分别对应不同的情况



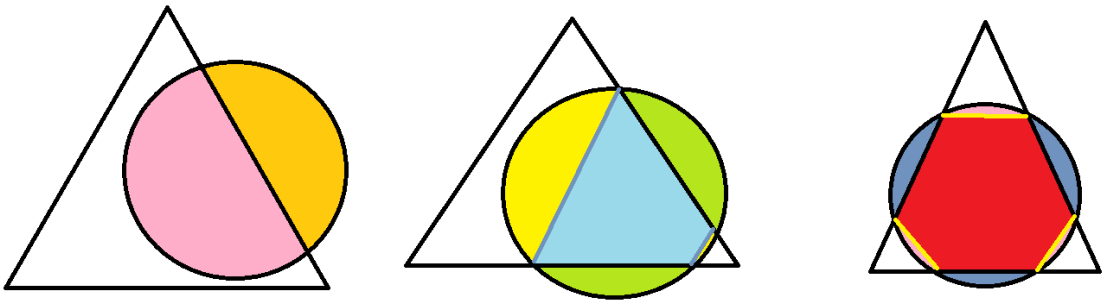


（这里也列出几种（其他类似），所求面积为$S_{橙色}+S_{粉色}$）

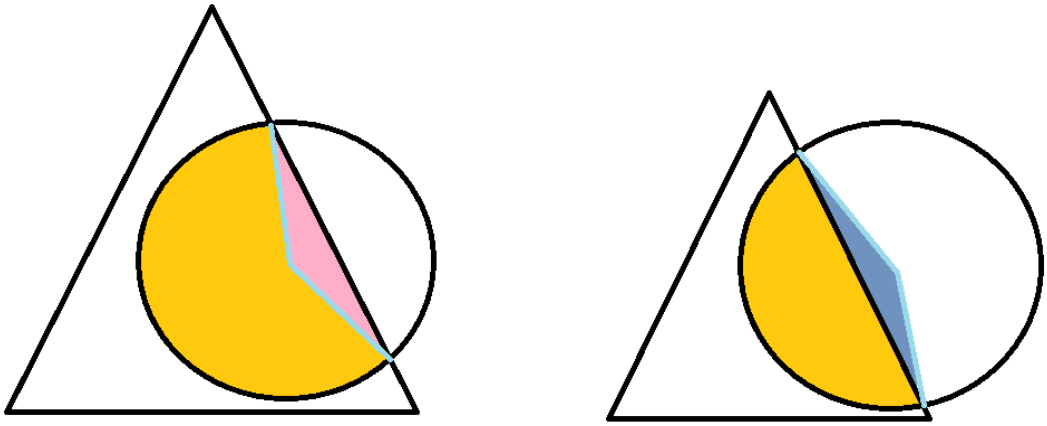
4. 三角形的三个顶点都在圆外，这个时候我们便会发现讨论的重点便由点在圆内的数量转化为边在圆内的数量，但不变的是我们仍需讨论圆心的位置

$$S=S_{优/劣弧}+S_{\Delta}$$

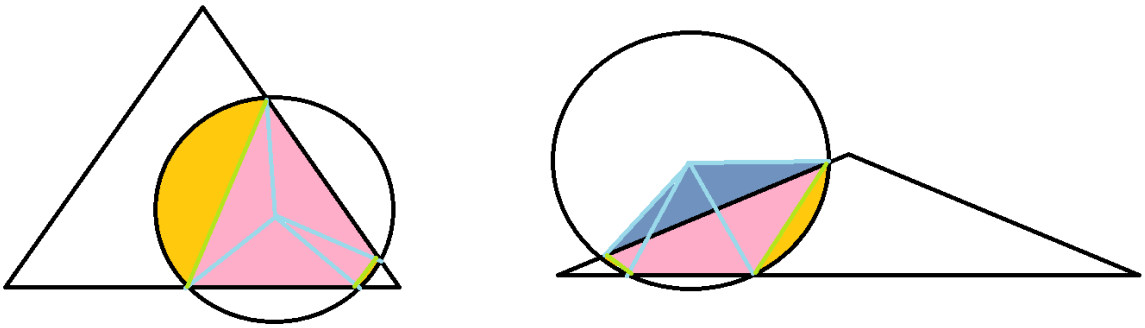
这个时候便需要通过在圆内的边数$(0,1,2,3)$分类（下图未列出三边都在圆外的情况，这种情况三角形包裹全部的圆，无分类情况），但与以上分类不同的是，在三边都在圆内的时候，可以发现不会出现圆心在粉色区域的情况，发现这个可以帮我们简化一次工作



对于仅1条边在圆内：

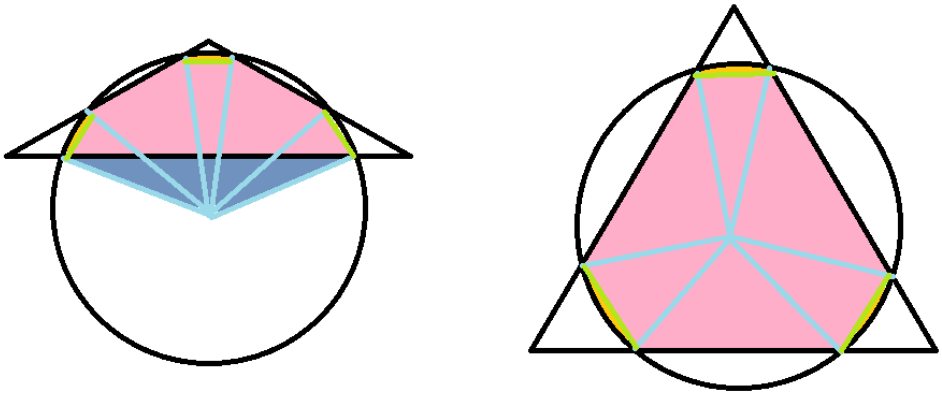


对于两条边在圆内：



（上图也列出了几种（其他类似），所求面积为橙色+粉色，深蓝色部分即为需要减去的部分）

对于三条边都在圆内：



然后根据上述对于情况的分类，我们便可以对每种情况分类求解了，可以一提的是，先对点距圆心的距离排序还有三个顶点在圆外时圆心对于边的距离进行排序后在进行计算相应的面积比较易于我们处理

此外，因为这个题目的数据量很小，不会卡时间，所以我们还可以将一些常用的运算封装成一个函数，可以减少很大的代码量，并且还可以帮我们避免无意间犯的错误

完结，撒花~

示例代码

```
#include<stdio.h>
#include<math.h>
#define pi 3.1415926535
#define eps 1e-6
double norm(double x,double y){return sqrt(x*x+y*y);}
double sign(double x){return x<0?-1.0:1.0;}
void swap(double *x,double *y){double tmp=*x;*x=*y;*y=tmp;}
double chordarea(double d,double r){
    if(d>=r)return 0;
    else {
        double u=d/r;
        return (r*r*(acos(u)-u*sqrt(1-u*u)));
    }
}
int main(){
    int n,pot_in_out;
    double r,x1,y1,x2,y2,x3,y3,area;
    scanf("%d",&n);
    while(n--){
        scanf("%lf %lf %lf %lf %lf %lf %lf",&r,&x1,&y1,&x2,&y2,&x3,&y3);
        double rc2=r*r,Ac=pi*r*r;
        double vl_x,vl_y,vu_x,vu_y,bl,al_x,al_y,bu,au_x,au_y,alv_x,alv_y,auv_x,auv_y;
        double At1,At2,At3,Aw;
        if(norm(x1,y1)>norm(x2,y2))swap(&x1,&x2),swap(&y1,&y2);
        if(norm(x1,y1)>norm(x3,y3))swap(&x1,&x3),swap(&y1,&y3);
        if(norm(x2,y2)>norm(x3,y3))swap(&x2,&x3),swap(&y2,&y3);
        double vminhat_x=x1/norm(x1,y1),vminhat_y=y1/norm(x1,y1);
        if(norm(x1,y1)>=r&&x2*vminhat_x+y2*vminhat_y>=r&&x3*vminhat_x+y3*vminhat_y>=r){
            area=0.00;
            pot_in_out=3;
        }
        else {
            int vf=(norm(x1,y1)>r)+(norm(x2,y2)>r)+(norm(x3,y3)>r);
            double k2,k3,b32_x,b32_y;
            double gll,guu,vll_x,vll_y,vuu_x,vuu_y,Aww,Att;
            double vd_x,vd_y,b1,b2,b3,nq1,nq2,nq3;
            double cp1_x,cp1_y,cp2_x,cp2_y,cp3_x,cp3_y;
            double A1,A2,A3;
            int nch;
            switch (vf){
                case 0:
                    pot_in_out=0;
                    area=0.5*fabs((x1-x3)*(y2-y3)-(y1-y3)*(x2-x3));
                    break;
                case 1:
```

```
                    pot_in_out=1;
                    vl_x=x3-x1,vl_y=y3-y1;
                    vu_x=x3-x2,vu_y=y3-y2;
                    bl=(sqrt((x3*vl_x+y3*vl_y)*(x3*vl_x+y3*vl_y)+(rc2-x3*x3-y3*y3)*
(vl_x*vl_x+vl_y*vl_y))-vl_x*x3-vl_y*y3)/(vl_x*vl_x+vl_y*vl_y);
                    al_x=x3+bl*vl_x,al_y=y3+bl*vl_y;
                    bu=(sqrt((x3*vu_x+y3*vu_y)*(x3*vu_x+y3*vu_y)+(rc2-x3*x3-y3*y3)*
(vu_x*vu_x+vu_y*vu_y))-vu_x*x3-vu_y*y3)/(vu_x*vu_x+vu_y*vu_y);
                    au_x=x3+bu*vu_x,au_y=y3+bu*vu_y;
                    alv_x=al_x-x3,alv_y=al_y-y3;
                    auv_x=au_x-x3,auv_y=au_y-y3;
                    At1=0.5*fabs(alv_x*auv_y-alv_y*auv_x);
                    At2=0.5*fabs(al_x*au_y-al_y*au_x);
                    At3=0.5*fabs(vl_x*vu_y-vl_y*vu_x);
                    Aw=0.5*acos((al_x*au_x+al_y*au_y)/rc2)*rc2;
                    area=Aw-At2+At3-At1;
                    break;
                case 2:
                    pot_in_out=2;
                    vl_x=x2-x1,vl_y=y2-y1;
                    vu_x=x3-x1,vu_y=y3-y1;
                    bl=(sqrt((x1*vl_x+y1*vl_y)*(x1*vl_x+y1*vl_y)+(rc2-x1*x1-y1*y1)*
(vl_x*vl_x+vl_y*vl_y))-vl_x*x1-vl_y*y1)/(vl_x*vl_x+vl_y*vl_y);
                    bu=(sqrt((x1*vu_x+y1*vu_y)*(x1*vu_x+y1*vu_y)+(rc2-x1*x1-y1*y1)*
(vu_x*vu_x+vu_y*vu_y))-vu_x*x1-vu_y*y1)/(vu_x*vu_x+vu_y*vu_y);
                    al_x=x1+bl*vl_x,al_y=y1+bl*vl_y;
                    au_x=x1+bu*vu_x,au_y=y1+bu*vu_y;
                    alv_x=al_x-x1,alv_y=al_y-y1;
                    auv_x=au_x-x1,auv_y=au_y-y1;
                    At1=0.5*fabs(alv_x*auv_y-alv_y*auv_x);
                    At2=0.5*fabs(al_x*au_y-al_y*au_x);
                    Aw=0.5*acos((al_x*au_x+al_y*au_y)/rc2)*rc2;
                    area=Aw-At2+At1;
                    b32_x=(x3-x2)/norm(x3-x2,y3-y2),b32_y=(y3-y2)/norm(x3-x2,y3-y2);
                    k3=x3*b32_x+y3*b32_y;
                    if(sqrt(x3*x3+y3*y3-k3*k3)<r){
                        k2=x2*b32_x+y2*b32_y;
                        gll=k3-sign(k3)*sqrt(k3*k3-(x3*x3+y3*y3-rc2));
                        guu=-k2+sign(k2)*sqrt(k2*k2-(x2*x2+y2*y2-rc2));
                        vll_x=x3-gll*b32_x,vll_y=y3-gll*b32_y;
                        vuu_x=x2+guu*b32_x,vuu_y=y2+guu*b32_y;
                        Aww=0.5*acos((vll_x*vuu_x+vll_y*vuu_y)/rc2)*rc2;
                        Att=0.5*fabs(vll_x*vuu_y-vll_y*vuu_x);
                        area=area-Aww+Att;
                    }
                    break;
                case 3:
                    pot_in_out=3;
                    vd_x=x3-x2,vd_y=y3-y2;
                    b1=(x3*vd_x+y3*vd_y)/(vd_x*vd_x+vd_y*vd_y);
                    cp1_x=x3-b1*vd_x,cp1_y=y3-b1*vd_y;
                    nq1=norm(cp1_x,cp1_y);
                    vd_x=x3-x1,vd_y=y3-y1;
                    b2=(x3*vd_x+y3*vd_y)/(vd_x*vd_x+vd_y*vd_y);
                    cp2_x=x3-b2*vd_x,cp2_y=y3-b2*vd_y;
                    nq2=norm(cp2_x,cp2_y);
```

```
                vd_x=x2-x1,vd_y=y2-y1;
                b3=(x2*vd_x+y2*vd_y)/(vd_x*vd_x+vd_y*vd_y);
                cp3_x=x2-b3*vd_x,cp3_y=y2-b3*vd_y;
                nq3=norm(cp3_x,cp3_y);
                if(norm(cp1_x,cp1_y)>norm(cp2_x,cp2_y)){
                    swap(&b1,&b2),swap(&nq1,&nq2);
                    swap(&cp1_x,&cp2_x),swap(&cp1_y,&cp2_y);
                }
                if(norm(cp1_x,cp1_y)>norm(cp3_x,cp3_y)){
                    swap(&b1,&b3),swap(&nq1,&nq3);
                    swap(&cp1_x,&cp3_x),swap(&cp1_y,&cp3_y);
                }
                if(norm(cp2_x,cp2_y)>norm(cp3_x,cp3_y)){
                    swap(&b2,&b3),swap(&nq2,&nq3);
                    swap(&cp2_x,&cp3_x),swap(&cp2_y,&cp3_y);
                }
                nch=(nq1<r)+(nq2<r)+(nq3<r);
                area=0.0;
                switch(nch){
                    case 0:
                        if(cp1_x*cp2_x+cp1_y*cp2_y<=0)area=Ac;
                        break;
                    case 1:
                        if(fabs(b1)>1)area=0.0;
                        else {
                            if(cp1_x*cp2_x+cp1_y*cp2_y<0)area=Ac-chordarea(nq1,r);
                            else area=chordarea(nq1,r);
                        }
                        break;
                    case 2:
                        A1=A2=0.0;
                        if(fabs(b1)<=1)A1=chordarea(nq1,r);
                        if(fabs(b2)<=1)A2=chordarea(nq2,r);
                        if(cp1_x*cp2_x+cp1_y*cp2_y<0)area=Ac-A1-A2;
                        else area=A1-A2;
                        break;
                    case 3:
                        A1=A2=A3=0.0;
                        if(fabs(b1)<=1)A1=chordarea(nq1,r);
                        if(fabs(b2)<=1)A2=chordarea(nq2,r);
                        if(fabs(b3)<=1)A3=chordarea(nq3,r);
                        if(cp1_x*cp2_x+cp1_y*cp2_y<0)area=Ac-A1-A2-A3;
                        else area=A1-A2-A3;
                        break;
                    default:
                        break;
                }
                break;
            default:
                break;
        }
    }
    double ans=area/Ac;
    printf("%.2lf %d\n",ans,pot_in_out);
    if(ans<0.20||ans>0.80)printf("Damn Sheep, come back tomorrow!\n");
    else printf("Well, maybe not too bad\n");
```

```
    }
    return 0;
}
```