2014 级 C++第三次练习解题报告

14211067 朱福

(A)新安当的神笔

描述

测试说明与提交

题目描述：

新安当这天来了个稀客，没错就是传说中没节操水平超越了延帝的天臧散人，Arthur。
老板景天感到非常荣幸，忍痛拿出了李三思家祖传的铁观音来请 Arthur，作为答谢，Arthur 掏出一支狼毫，在地上画了个惟妙惟肖的菱形，唰的一闪，地上出现了一枚等大的钻石。
看着呆若木鸡口水直流的景天，Arthur 微微一笑“景老板请吧……”

输入：

一组测试数据。
第一行为一个整数 n(0≤n≤20)，表示钻石的数目.
接下来 n 行，每一行一个整数，为每块钻石的尺寸。

输出：

对于每个尺寸，输出对应的钻石，用*填充，具体形式参见输出样例。

输入样例：

3
2
3
5

输出样例：

```
 *
***
 *
  *
 ***
*****
 ***
  *
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

解析：本题考察循环语句的应用。
易错点：当输入整数 n 时，输出的菱形对角线上应该有(2*n-1)个”*”，不要误以为是 n 个。

参考程序：

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int t, n;
    cin >> t;//t为钻石数目
    while (t--)
    {
        cin >> n;
        for (int i = -n + 1; i <= n - 1; i++)
        {
            for (int j = -n + 1; j <= n - 1; j++)
            {
                if (abs(i) + abs(j) <= n - 1)
                    //符合该语句的点恰好组成菱形
                    cout << "*"; else cout << " ";
            }
            cout << endl;
        }
    }
    return 0;
```

}

(B)逃出迷宫

Problem Description

小蛮和龙幽两人在江湖上行侠仗义，锄强扶弱。老百姓非常感谢他们，但他们同时也结下了不少仇家。很明显，这些仇家都不是什么见得了人的人物，干得全都是下流手段。某天他们向小蛮和龙幽下了战帖，说要来一场光明正大的决斗，如果他们赢了，以后小蛮和龙幽就不得再干涉他们的事。龙幽觉得其中有诈，刚想拒绝。可是任性的小蛮却一口答应了下来，龙幽（一 一+）……
等他们到了约定的地方却发现对方没有按时到，龙幽敏锐的察觉到事情不对，拉着小蛮就要离开，可这时虚空中传来了声音：“既来之，则安之。何必这么急着走呢……这个阵法是专门为你们留的，找不到破阵之法，你们就将永远被困在里面了，哈哈哈哈……”小蛮大声斥责他们无赖，龙幽无奈的回答“他们本来就是无赖(ー.ー|||，我们还是想办法出去吧”说完开始研究这个阵法……
经过一番研究，龙幽发现此阵很奇特，每一步都得走特定的步数，一步错则得从头开始。经过一番尝试，他发现前几步满足下面这个数列……
第一项：1 第二项：2 第三项：5 第四项：26 第五项：677••••••
现在请你帮助他们逃出这座迷阵吧……

Input

输入包含多组数据，每组数据为一行，为一个整数 n（1<=n<=100）。

Output

对于每一个 n，输出他们第 n 步需要走的步数。现在只要求你给出这个步数的十位数的值。

Sample Input

3
5
6

Sample Output

0
7
3

解析：这一题也是使用循环，很容易找到规律：第(n+1)步要走的步数总是第 n 步步数的平方再加一。
易错点：平方很容易让数据超出范围，其实因为题目只要求输出第十位，将每次步数只保留后两位即可。

参考程序：

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
{
    int t;
    while (cin >> t)
    {
        int A[2]; A[1] = 1;
        if (t <= 3)
        {
            cout << "0\n"; continue;//步数不大于3时，十位数是0
        }
        for (int i = 2; i <= t; i++)
        {
            A[i % 2] = (A[(i + 1) % 2] * A[(i + 1) % 2] + 1) % 100;
            //对100取模，只保留后两位
        }
        cout << A[t % 2] / 10 << endl;
    }
    return 0;
}
```

题目描述

第 X 次圣杯战争即将拉开，作为参与其中的魔术师之一，Darling 首先要做的，是绘制召唤法阵。
已知绘制法阵需要若干种化学元素，原本这对于 Darling 并非什么难事，但是 Darling 近期有点忙，因此她来找你帮忙。
你的任务很简单，给你所有原子序数，请你将 Darling 需要的元素名称写下来并交给化学药品店。

输入

一组测试数据。
第一行包含一个数 n（1≤n），表示 Darling 绘制法阵需要的元素个数。
第二行包含 n 一个整数，用空格隔开，为每种元素对应的原子序数。

输出

输出一行，为所需的全部元素名称（化学符号），元素与元素之间用一个空格隔开。（如果元素不在前四周期，则该元素名称为 Usl）。
具体参见输出样例

输入样例

3
1 17 50

输出样例

H Cl Usl

解析：本题只需要 switch 语句即可，顺便帮大家复习了高中化学。

易错点：前四周期（H~Kr）共 36 种元素，超出的输出 Usl。

参考程序：

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
```

```
{
    int s,t;
    while (cin >> t)
    {
        for (int i = 0; i < t; i++)
        {
            cin >> s;
            switch (s)//判断原子序数
            {
            case 1:
                cout << "H "; break;
            case 2:
                cout << "He "; break;
            case 3:
                cout << "Li "; break;
            case 4:
                cout << "Be "; break;
            case 5:
                cout << "B "; break;
            case 6:
                cout << "C "; break;
            case 7:
                cout << "N "; break;
            case 8:
                cout << "O "; break;
            case 9:
                cout << "F "; break;
            case 10:
                cout << "Ne "; break;
            case 11:
                cout << "Na "; break;
            case 12:
                cout << "Mg "; break;
            case 13:
                cout << "Al "; break;
            case 14:
                cout << "Si "; break;
            case 15:
                cout << "P "; break;
            case 16:
                cout << "S "; break;
            case 17:
                cout << "Cl "; break;
            case 18:
```

```
                cout << "Ar "; break;
            case 19:
                cout << "K "; break;
            case 20:
                cout << "Ca "; break;
            case 21:
                cout << "Sc "; break;
            case 22:
                cout << "Ti "; break;
            case 23:
                cout << "V "; break;
            case 24:
                cout << "Cr "; break;
            case 25:
                cout << "Mn "; break;
            case 26:
                cout << "Fe "; break;
            case 27:
                cout << "Co "; break;
            case 28:
                cout << "Ni "; break;
            case 29:
                cout << "Cu "; break;
            case 30:
                cout << "Zn "; break;
            case 31:
                cout << "Ga "; break;
            case 32:
                cout << "Ge "; break;
            case 33:
                cout << "As "; break;
            case 34:
                cout << "Se "; break;
            case 35:
                cout << "Br "; break;
            case 36:
                cout << "Kr "; break;
            default:
                cout << "Usl "; break;
            }
        }
        cout << endl;
    }
    return 0;
```

}

(D)盗墓笔记之禁婆的考验

题目描述

推开那扇坑爹的石门，忽然一只冷冰冰的手搭在了 Thor 的肩上。回头，一张脸惊得他毛骨悚然——居然是——Arthur？！
长着一张 Atrhur 的脸的禁婆露出一个妩媚的微笑。
想过我这一关？先来玩个小游戏吧。很简答，给你一个十进制的数字 n，你能求出它在 m 进制下的表示里数字 k 的个数么？
于是加法与求模都不会做的 Thor 又逗比了，怎么办？看你喽~

输入

多组测试数据。
每组数据为一行，包含三个整数 n,m,k（2≤m≤8，0≤k≤9）。保证 n 在 int 范围内。

输出

对于每组数据，输出一行，包含一个整数，为 k 的个数。

输入样例

9 2 1
8 8 1

输出样例

2
1

解析：这一题需要通过循环求出 m 进制下数字 k 出现次数。

易错点：要注意循环结束的标志。

参考程序：

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int n, m, k, a;
    while (cin >> n >> m >> k)
    {
        a = 0;
        while (n != 0)//当n不为0时
        {
            if (n%m == k)a++;
            //判断该位数字是否是k
            n /= m;
        }
        cout << a << endl;
    }
    return 1;
}
```

(E)盗墓笔记之怒海潜沙

题目描述

西沙水下，一阵暗流，Thor 与天真一行人被冲散了。
幸运的是，他被冲到了一扇门前。
不幸的是，刚刚到达这扇门前，门就合上了。上面的机关被重置为初始状态。
经过仔细的观察，Thor 发现，机关为一个圆盘，分为内外两圈。内圈均匀刻着 a 个小格，外圈均匀刻着 b 个小格，每个刻度有一个独一无二的图腾与之对应。
他还发现，每经过一分钟，内外两圈会同时顺时针旋转一个单位。从身边墙壁上的铭文，他

得知，当轮盘再次回到初始状态时，门会重新打开。他想知道从门被合上到再次打开的这一个轮回需要多少时间，这个任务就交给你了。

输入

多组测试数据。
对于每组测试数据，输入两个数 a,b（1≤a，b≤30000），用空格隔开，具体含义见题目描述。

输出

对于每组数据，输出一行，包含一个整数，表示一个轮回需要的时间。

输入样例

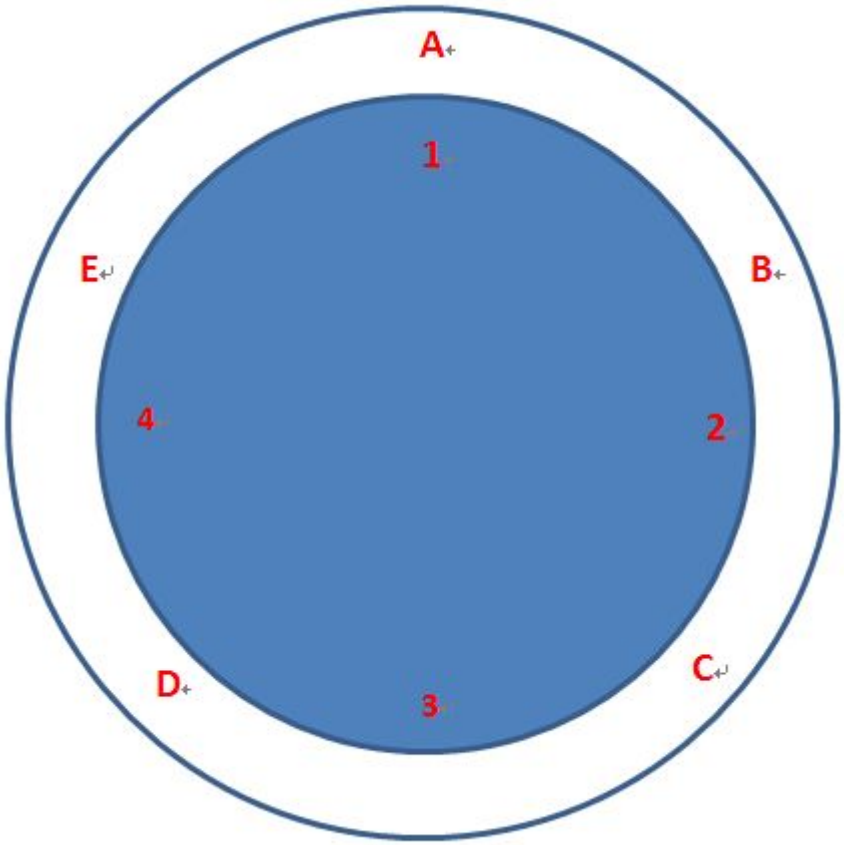
3 4
4 8
12 16

输出样例

12
8
48

HINT

机关？大概.....就是这个样子吧.....OTL



解析：只用求出 a，b 的最小公倍数即可，使用循环的方法。

易错点：如果从 1 至 a*b 逐一判断太过麻烦，事实上我们根据一个数学规律：最小公倍数乘以最大公约数等于两数之积，可以在较短时间内得出答案。

参考程序：

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int n, m, a, mm, nn;
    while (cin >> n >> m)
    {
        mm = m; nn = n;
        while (n != 0 && m != 0)
        {//先求出最大公约数
            a = n%m;
            n = m;
            m = a;
```

```
        }
        cout << mm*nn / (m + n) << endl;
        //(m+n)是最大公约数
    }
    return 1;
}
```

(F)盗墓笔记之逃亡

题目描述

电闪雷鸣，警笛大作——不明人士的举报使得乘着列车去往长白雪山的 Thor 与天真不得不搞了一次夜半逃亡之旅。给你一上帝视角，将他们跑路的路线与警方的追查路线简化为一平面直角坐标系上的两条直线，由于警方人手的优越性，只要 Thor 与天真有可能出现在对方的路线上，就会被包围逮捕。他们只能找一条完全与警方没有交集的路线，才可以安全逃离。现请给你这两条路线的一般表示（Ax+By+C=0），请你判断他们是否会遭到逮捕。

输入

多组测试数据。
每组测试数据分为两行。第一行包含三个整数 A1,B1,C1，用空格隔开，为 Thor 和天真的逃跑路线。第一行包含三个整数 A2,B2,C2，用空格隔开，为警方的追捕路线。
保证输入数据可以构成直线。

输出

对于每组数据，输出一行。如果 Thor 与天真会被逮捕，输出“Dead End”。如果他们安全逃离，则输出“Safe and Sound”

输入样例

1 1 1
1 1 2
1 1 1
1 2 1

输出样例

Safe and Sound
Dead End

解析：题意很明确，判断两直线是否相交。

易错点：两直线重合的情况容易忽略，而这时无法安全逃离。另外，数据类型一定要选 **long long** 型，否则精度不够。

参考程序：

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    long long a, b, c, a1, b1, c1;
    while (cin >> a >> b >> c >> a1 >> b1 >> c1)
    {
        if (a*b1 == a1*b&&a*c1 != c*a1)
            //两直线平行且不重合
            cout << "Safe and Sound" << endl;
        else //两直线相交
            cout << "Dead End" << endl;
    }
    return 1;
}
```

(G) 李逍遥的仙剑客栈

Description

有一天天臧散人 Arthur 到了渝州东南的仙剑客栈，发现李逍遥这小子粗心大意在上酒的时候，有的桌上多上了酒，有的桌上没上酒。唉，谁让 Arthur 心软呢，不忍心看李逍遥被他婶婶骂，决定帮他——收拾一排桌子。
给你一个数 n 表示有多少个桌子，接下来给你一段整数序列表示每个桌上需要的酒量（假设这些桌子在一条直线上，且每个桌子之间的距离都是 1），正数表示多放了几瓶酒，负数表示应该放多少瓶酒。请你帮 Arthur 算一下他提着酒走的最短路程是多少。对了，Arthur 体力太渣，一次只能拿一瓶酒。

Input

第一行一个数 T 表示有 T 组数据。
接下来 T 组数据，每组数据有 2 行。
第一行一个数 n(1<=n<=1000)，表示桌子数量，
接下来第二行有一段数列 Ai(|Ai|<=1000)，表示每个桌子上应该放的酒。保证数列总和为 0。

Output

对于每组测试数据，输出一个数，表示 Arthur 拿着酒走的最少总路程。

Sample

Input：

```
2
3
-1 2 -1
6
-1 -1 -1 3 -1 1
```

Output：

```
2
7
```

Hint

其实就是每瓶酒走的路程之和。

解析：从少放了酒的桌子入手，寻找最近的且多上酒的桌子，以多补少。

易错点：如何寻找最近的多上酒的桌子是关键，其实只需在循环中总是从 1 到 n 即可。

参考程序：

```
#include<iostream>
using namespace std;
int main()
{
    int t, n, A[1001];
    cin >> t;
    while (t--)//共t组数据
    {
        cin >> n;
        for (int i = 1; i <= n; i++)
            cin >> A[i];
        int s = 0;
        for (int i = 1; i <= n; i++)
        {
            if (A[i] < 0)
            {
                for (int j = 1; j <= n; j++)
                {
                    if (A[j] > 0)
                    {
                        int count = (-A[i]) < A[j] ? (-A[i]) : A[j];
                        //该次转移的酒的瓶数
                        int temp = (i - j) > (j - i) ? (i - j) : (j - i);
                        //对(i-j)取绝对值
                        s = s + count*temp;
                        //移动酒的路程
                        A[i] += count; A[j] -= count;
                        if (A[i] == 0)break;
                    }
                }
            }
        }
        cout << s << endl;
    }
    return 0;
}
```

(H) jhljx 学函数

Problem Description

jhljx 听说大家学了函数，决定考察大家的基本功。
给你两个数 a 和 b，请用函数来实现交换这两个数，使得 a 的值为 b,b 的值为 a。
本题必须用函数来完成。
不要在函数中先输出 b,再输出 a。保证 a 的值是 b，b 的值是 a。请不要水过。

Input

输入多组数据。
每组数据一行，为两个数 a 和 b。(a 和 b 在 int 范围内)

Output

输出进行交换后 a 和 b 的值。

Sample Input

1 2

Sample Output

2 1

Hint

函数声明的方法：

方法 1

```
int fuc(int);
int main()
{

}
int fuc(int a)
{
}
```

方法 2

```
int fuc(int a)
{
}
int main()
{
}
```

关于函数的值传递和引用传递

值传递

值传递是将数值传递给一个函数，但是函数中得到的数值只是原数值的一个副本。函数中对它进行操作，不会改变 main 函数中传递进来的那个参数的实际值。

```
void fuc(int m,int n)
{
m++;n++;
cout<<m<<" "<<n<<endl;
}
int main()
{
int a,b;
cin>>a>>b;
fuc(a,b);
cout<<a<<" "<<b<<endl;
}
```

如果输入 2 3，输出第一行为 3 4，第二行是 2 3。第一行的 3 4，是在函数中进行运算的结果。而第二行的 2 3，是原来 a,b 的值，说明调用 fuc 函数进行值传递时，没有改变原有 a 和 b 的值。

引用传递

引用传递是将数值传递给一个函数，函数中对数值进行操作会改变原来的值。

```
void fuc(int &m,int &n)
{
m++;n++;
cout<<m<<" "<<n<<endl;
}
int main()
{
int a,b;
cin>>a>>b;
fuc(a,b);
cout<<a<<" "<<b<<endl;
```

}
如果输入 2 3，就会输出两行 3 4。这说明在函数中 a,b 原来的值就已经改变了。

解析：考察函数的按值传递与引用传递的区别
注意点：此题需要用引用传递

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
void f(int &a, int &b)//此处为引用传递
{
    int t = a; a = b; b = t;
}
int main()
{
    int t, s;
    while (cin >> t >> s)
    {
        f(t, s);
        cout << t << " " << s<< endl;
    }
    return 0;
}
```

（I*）Ryan's Derivatives

Problem Description

Ryan 最近在阮姐的课上学习了导数= =

有一天阮姐大笔一挥写下了一个多项式：

$$f(x) = a_0x^{b_0} + a_1x^{b_1} + \cdots + a_{n-1}x^{b_{n-1}} + a_nx^{b_n}$$

然后又大笔一挥写下了另一个东西：

$$\frac{d^m f(x)}{dx^m}$$

然后说：求出它的表达式。这个当作课后作业= =

数学很烂的 Ryan 怎么能禁受得住阮姐的轰炸，只好让你来帮助他完成阮姐的课后作业。

Input

多组测试数据。

每组测试数据包含 n＋1 行。

第一行为两个整数 n, m

接下来 n 行，每行两个整数，ai 和 bi

对于每组数据，所有 bi 非负且均不相同

Output

对于每组数据，输出一行，为 f(x)的 m 次导数的表达式。

所有系数和指数应符合多项式书写的基本原则。

如：5 倍的 x 四次方 表示为 5x^4

1 倍的 x 不应表示为 1x^1 而是 x

5x^3+0x^2+(-x)应表示为 5x^3-x

输出结果各项顺序与输入一致

Sample Input

```
4 2
1 4
2 3
3 2
4 1
3 1
1 5
-1 1
1 0
```

Sample Output

```
12x^2+12x+6
5x^4-1
```

解析：在输入数据时，如果某项的次数小于求微分的次数时，该项求微分后一定为 **0**，所以可以直接忽略。

易错点：输出时，要符合基本规范，次数由高到低，如果首项系数大于 **0**，则不需要输出加号。

参考程序：

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int n, m;
    while (cin >> n >> m)
    {
        int A[1000], B[1000], num = 0, flag = 0;
        for (int i = 0; i <n; i++)
        {
            cin >> A[num] >> B[num];
            B[num] -= m;
            if (B[num]<0)num--;
            //如果某项的次数小于求微分的次数时,忽略该项
            num++;
        }
        if (num == 0)flag = 1;
        for (int i = num - 2; i >= 0; i--)
        {//按次数由大到小排序
            int mi = i;
            for (int j = num - 2; j >= 0; j--)
                if (B[j] < B[j + 1])
                    mi = i;
            int tp = B[mi]; B[mi] = B[i]; B[i] = tp;
            tp = A[mi]; A[mi] = A[i]; A[i] = tp;
        }
        for (int i = 0; i <num; i++)
        {
```

```
            for (int j = B[i] + m; j > B[i]; j--)
                A[i] *= j;//求导后的系数
            if (i != 0 && A[i] > 0)cout << "+";
            cout << A[i];
            if (B[i] == 1)cout << "x";//次数为1
            if (B[i] > 1)cout << "x^" << B[i];//次数大于1
        }
        if (flag)cout << 0;
        cout << endl;
    }
    return 0;
}
```

（J**）RecTangles！RecTangles！RecTangles！

题目描述

Arthur 这次也准备走一次简单路线，于是题面很简单。
给你平面直角坐标系下两个矩形中位于对角线上两个顶点的坐标，请判断这两个矩形是否会相互重叠。
矩形的两边与坐标轴平行，也就是说不会出现斜置的矩形。

输入

多组测试数据，每组数据为两行。
对于每组数据，第一行为四个整数，表示第一个矩形的两个对角顶点的坐标（x1,y1）与(x2,y2)。
第二行为四个整数，表示第二个矩形的两个对角顶点的坐标（x3,y3）与(x4,y4)。
(0≤x1,y1,x2,y2,x3,y3,x4,y4≤100)

输出

对于每组数据，如果相交，则输出“YES”，否则输出“NO”

输入样例

0 0 2 2
1 1 3 3
0 0 0 1
1 1 1 2

输出样例

YES
NO

HINT

重叠一个顶点或者一条边也算重叠
请考虑矩形合理性

解析：本题我的想法是将矩形横(纵)坐标看成闭区间，这样就转为高中的集合问题了。

易错点：重叠点或边也算重叠，而且矩形边总大于 0.

参考程序：

```
#include<iostream>
using namespace std;
int main()
{
    int a, b, c, d, e, f, g, h, s, t;
    while (cin >> a >> b >> c >> d >> e >> f >> g >> h)
    {
        t = a < c ? a : c; s = a;
        a = t;
        c = s + c - t;

        t = b < d ? b : d; s = b;
        b = t;
        d = s + d - t;

        t = e < g ? e : g; s = e;
        e = t;
        g = s + g - t;

        t = f < h ? f : h; s = f;
        f = t;
        h = s + h - t;
```

```
        if (a <= g&&c >= e&&b <= h&&d >= f
            //矩形1的左侧横坐标不大于矩形2的右侧横坐标
            //矩形1的右侧横坐标不小于矩形2的左侧横坐标，其他同理
            &&a != c&&b != d&&e != g&&f != h)
            //矩形各边大于0
            cout << "YES\n";
        else cout << "NO\n";
    }
    return 0;
}
```

(K***) Talus 的会面

Description

国庆节将至，Talus 决定去好好领略一下祖国的大好山川，早早就安排好了行程。最近他获知了这样一个消息，一个多年未见的朋友恰好也经过 Talu 乘坐的火车途中经过的某一站，简称为 A 站。

但是由于 Talus 与朋友的行程还并未完全确定，所以 Talus 会在[t1,t2]中的任意时刻以相同的概率密度到 A 站，同样的，他的朋友则会在[s1,s2]之间以相同的概率密度到达 A 站。火车在 A 站停留的时间都是 W，只有在同一时刻，两人的火车都停在 A 站的时候，两人才能进行简短的会面，现在 Talus 想知道他和朋友能会面的概率是多大？

Input

第一行输入一个整数 T（T <= 12），表示测试数据组数

接下来每一行输入 t1,t2,s1,s2,w5 个整数(360<=t1,t2,s1,s2<=1080,1 <= w <= 90)，如题中描述。

Output

对于每组数据，输出一行，表示能会面的概率

Sample Input

```
2
1000 1040 1000 1040 20
```

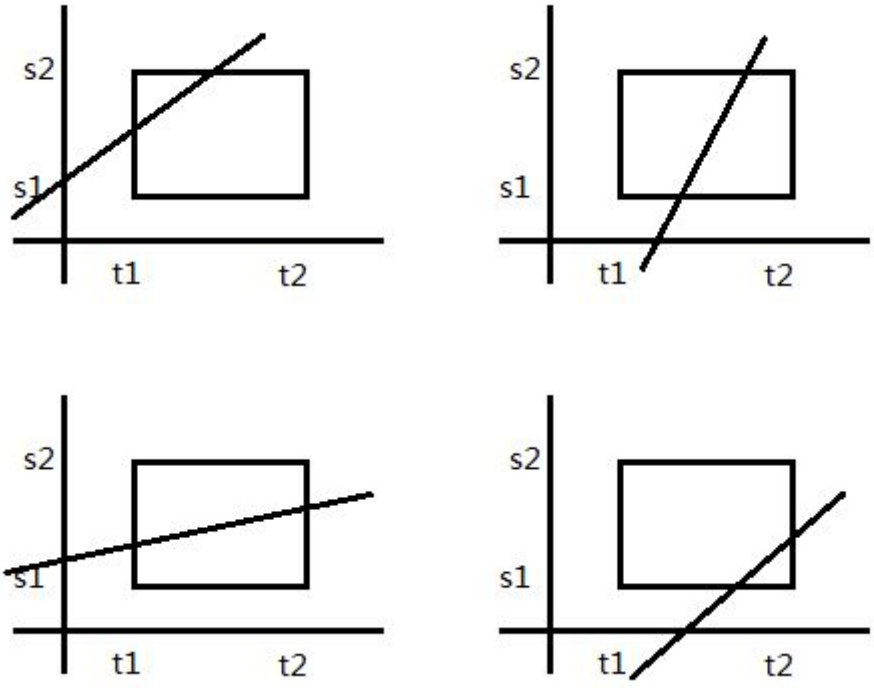
```
720 750 730 760 16
```

Sample Output

```
0.75
0.67
```

解析：这是高中时学过的概率题，通过线性规划来解决。先画出可行域，再画出符合条件的区域，后者比上前者即为结果。

易错点：要考虑直线与矩形相交的多种情况，每条线都有四种情况。



参考程序：

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
{
    int t1, t2, s1, s2, w, t;
    cin >> t;
    while (t--)
    {
        cin >> t1 >> t2 >> s1 >> s2 >> w;
        if (s2 - t1 <= 0 || t2 - s1 <= 0)
        {
```

```
    cout << "0.00" << endl; continue;
}//不可能会面
int rgn1 = -1, rgn2 = -1;
if (s2 <= t1 + w)rgn1 = 0;
if (s1 >= t2 - w)rgn2 = 0;
if (rgn1 == 0 && rgn2 == 0)
{
    cout << "1.00\n"; continue;
}//一定能会面
if (s2 > t1 + w)
{
    int line1 = 0, line2 = 0, x1, y1, x2, y2;

    //寻找第一条线与可行域矩形的交点
    for (int i = t1 + 1; i <= t2; i++)
        if (i + w == s2){ line1 = 1; x1 = i; y1 = s2; break; }
    for (int i = s1 + 1; i < s2; i++)
        if (t2 + w == i){ line1 = 2; x1 = t2; y1 = i; break; }
    for (int i = t1; i < t2; i++)
        if (i + w == s1){ line2 = 3; x2 = i; y2 = s1; break; }
    for (int i = s1; i < s2; i++)
        if (t1 + w == i){ line2 = 4; x2 = t1; y2 = i; break; }
    if (y1 == s2&&x2 == t1)
        rgn1 = (s2 - w - t1)*(s2 - w - t1);
    if (x2 == t1&&x1 == t2)
        rgn1 = (2 * s2 - 2 * w - t1 - t2)*(t2 - t1);
    if (y1 == s2&&y2 == s1)
        rgn1 = (s2 + s1 - 2 * w - 2 * t1)*(s2 - s1);
    if (x1 == t2&&y2 == s1)
        rgn1 = (s2 - s1)*(t2 - t1) * 2 - (y1 - s1)*(t2 - x2);
}
if (s1 < t2 - w)
{
    int x1, y1, x2, y2;

    //寻找第二条线与可行域矩形的交点
    for (int i = t1 + 1; i <= t2; i++)
        if (i - w == s2){ x1 = i; y1 = s2; break; }
    for (int i = s1 + 1; i < s2; i++)
        if (t2 - w == i){ x1 = t2; y1 = i; break; }
    for (int i = t1; i < t2; i++)
        if (i - w == s1){ x2 = i; y2 = s1; break; }
    for (int i = s1; i < s2; i++)
        if (t1 - w == i){ x2 = t1; y2 = i; break; }
```

```
            if (x1 == t2&&y2 == s1)
                rgn2 = (y1 - s1)*(t2 - x2);
            if (y1 == s2&&y2 == s1)
                rgn2 = (2 * t2 - x1 - x2)*(s2 - s1);
            if (x1 == t2&&x2 == t1)
                rgn2 = (y1 + y2 - 2 * s1)*(t2 - t1);
            if (y1 == s2&&x2 == t1)
                rgn2 = (s2 - s1)*(t2 - t1) * 2 - (x1 - t1)*(s2 - y2);

        }

        double temp = (double)(s2 - s1)*(t2 - t1) * 2;//可行域总面积

        double ps = (double)((temp - rgn1 - rgn2) / temp);//会面几率
        cout << fixed << setprecision(2) << ps << endl;
    }
    return 0;
}
```