A.收集普莱姆

解题思路

本题考察了找规律。

首先，10^{100} 意味着肯定不能暴力去做。通过观察样例+模拟一些更复杂的样例，可以发现，当操作次数趋近无限时，每一段若干个(记为x个)R盒和其右侧紧挨着的若干个(记为y个)L盒中的普莱姆会聚集到RL邻接处的两个盒中，即只有 "RL" 这样的相邻的盒子对中才会有滞留的普莱姆。并且通过观察会发现，如果x+y是偶数，最终两个邻接盒会分别滞留(x+y)/2个普莱姆，否则某个盒会比另外一个盒恰好多1个普莱姆（x是偶数时R盒多，否则L盒多）因此分段处理即可得到答案。

AC代码

```
#include <stdio.h>
#include <string.h>
#define MN 100007

char str[MN];
int ans[MN];
int main()
{
    int i = 0, j = 0, len;
    int cnt_r, cnt_l;
    scanf("%s", str);
    len = strlen(str);
    while (i <= len - 2 || j <= len - 2)
    {
        cnt_r = cnt_l = 0;
        for (i = j; str[i] == 'R'; i++)
            ++cnt_r;
        for (j = i; str[j] == 'L'; j++)
            ++cnt_l;

        if ((cnt_r + cnt_l) % 2 == 0)
            ans[i] = ans[i - 1] = (cnt_r + cnt_l) / 2;
        else
        {
            if (cnt_r % 2 == 0)
                ans[i - 1] = (cnt_r + cnt_l - 1) / 2, ans[i] = ans[i - 1] + 1;
            else
                ans[i - 1] = (cnt_r + cnt_l + 1) / 2, ans[i] = ans[i - 1] - 1;
        }
    }
    for (i = 0; i < len; i++)
        printf("%d ", ans[i]);
    return 0;
}
```

B.yuki舒做游戏

解题思路

本题考察了简单的数论知识——素数。

最小的素数是2，因此任何大于等于2的整数都可以写成某个素数的正整数倍。

因此如果 $x-y>=2$，则答案为YES，否则为NO。

AC代码

```
#include <stdio.h>
int main()
{
    long long x, y;
    while (scanf("%lld %lld", &x, &y) != EOF)
        printf(x-y >= 2 ? "YES\n" : "NO\n");

    return 0;
}
```

C.矩阵A*B problem

解题思路

本题考察了二维数组和多重循环。

大家在高等代数上都学过了矩阵乘法（书2.4章），因此我们出了这道题考察大家对二维数组和多重循环的掌握。

直接模拟进行矩阵乘法即可。

AC代码

```c
#include <stdio.h>
#define MN 222

long long a[MN][MN], b[MN][MN];

int main()
{
    int i, j, k, n, m, p;
    long long term;
    scanf("%d %d %d", &n, &m, &p);

    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            scanf("%lld", &a[i][j]);
    for (i = 0; i < m; i++)
        for (j = 0; j < p; j++)
            scanf("%lld", &b[i][j]);

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < p; j++)
        {
            term = 0;
            for (k = 0; k < m; k++)
                term += a[i][k] * b[k][j];
            printf("%lld ", term);
        }
        printf("\n");
    }

    return 0;
}
```

D.花朵数

解题思路

根据题目要求，遍历1e8以内所有数字，求出各个位数的n次方和比较即可，生成花朵数代码如下。

```
#include <stdio.h>
int main(){
    int num = 1, k,n, i, j;
    int a[10];
    for (; num < 100000000; ++num) {
        k = num, n = 0;
        while(k) {
            a[n++] = k % 10;
            k /= 10;
        }
        for (i = 0; i < n; ++i) {
            int p = 1;
            for (j = 0; j < n; ++j) {
                p *= a[i];
            }
            k += p;
        }
        if(k == num) printf("%d\n", k);
    }
}

```

E.时针和分针

解题思路

可将时间相加后，化作时针分针在表盘转过的角度进行计算，以圈为单位。注意到分针速度时分针的12倍，将时针角度加上时针分针角度差的$\frac{1}{11}$即可。注意转换回时:分:秒的格式时需要考虑浮点数误差和输出-0问题。

```c
#include <stdio.h>
#include <math.h>
#define eps 1e-12
double read(){
    int h, m, s;
    scanf("%d:%d:%d", &h, &m, &s);
    return (double)h + m / 60.0 + s / 3600.0;
}

int main(){
    double a, h, m;
    int t;
    scanf("%d", &t);
    while(t--){
        a = read() + read();
        h = a / 12 - floor(a /12);//时针角度
        m = a - floor(a);//分针角度
        if(m > h) m -= 1;
        h += (h - m) / 11;
        int hh = h * 12 + eps, mm = h * 60 * 12 + eps;
        double ss = (h * 60 * 12 - mm) * 60;
        if(ss < 0) ss = 0;
        printf("%02d:%02d:%010.7f\n", hh % 12, mm % 60, ss);
    }
}
```

F. 一维离散卷积

解题思路

结合有效时域，可得出$h(t)=\sum_{x=a}^{b}f(x)g(t-x)$，使用循环进行累加求和即可。代码实现时，可将有效时域作平移到自然数部分，将函数值存储至数组中，并使用函数实现表达式中的分段函数。

```c
#include <stdio.h>
int a, b, c, d;
int fn[20005], gn[20005];
int fx(int x){
    if(a <= x && x <= b) return fn[x - a];
    return 0;
}

int g(int x){
    if(c <= x && x <= d) return gn[x - c];
    return 0;
}

int main(){
    int n, t, q, ans, i;
    scanf("%d", &n);
    while(n--){
        scanf("%d%d",&a, &b);
        for (i = a; i <= b; ++i) {
            scanf("%d", &fn[i - a]);
        }
        scanf("%d%d",&c, &d);
        for (i = c; i <= d; ++i) {
            scanf("%d", &gn[i - c]);
        }
        scanf("%d", &t);
        while(t--){
            scanf("%d", &q);
            ans = 0;
            for (i = a; i <= b; ++i) {
                ans += fx(i) * g(q - i);
            }
            printf("%d ", ans);
        }
        putchar('\n');
    }
}
```

H. 直角坐标转极坐标

解题思路

由数学知识可知，将直角坐标转为极坐标，算法为：

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \begin{pmatrix} tan(y \div x)\ ,\ x \neq 0 \\ \pm\pi \div 2\ ,\ x = 0 \end{pmatrix}$$

在 $math.h$ 中有一个叫做 $atan2()$ 的函数，它是正切函数 $tan()$ 的变种之一，对于任意不同时等于0的参数 x 和 y ， $atan2(y,x)$ 所表达的意思是坐标原点为起点，指向 (x,y) 的射线在坐标平面上与 x 轴正方向之间的角的角度 。在几何意义上，$atan2(y,x)$ 等价于 $atan(y/x)$，但 *atan2* 的最大优势是可以正确处理 $x = 0$ 而 $y \neq 0$ 的情况，而不必进行会引发除零异常的 y/x 操作。

该函数定义域如下：

$$\mathbf{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

需要特别注意的是，$atan2()$ 函数的返回值范围是 $(-\pi, \pi]$ ，而本题要求的是$[0, 2\pi)$

AC代码

```cpp
#include<stdio.h>
#include<math.h>
int main()
{
    int x, y;
    double pi = acos(-1.0);
    while (~scanf("%d%d", &x, &y)) {
        double r = sqrt(x * x + y * y);
        double theta = atan2(y, x);
        if(theta<0) theta+=2*pi;
        printf("%.3lf %.3lf\n", r, theta);
    }
    return 0;
}
```

G. Ganten学姐的cosx

解题思路

根据 Hint 这道题不能用 $math.h$ 中的 $cos()$ 函数，需要自己写代码对 cos 级数求和。

本题只保留8位小数，故当 cos 级数和的前 m 项和前 $m+1$ 项之间的差小于 $1e-8$ 即可判定，对于本题的输出结果不再发生变化，可以停止计算。

AC代码

```
#include "stdio.h"
#include "math.h"
long long factorial(int n)
{
    int i;
    long long ans=1;
    for(i=1;i<=n;i++)
    {
        ans = ans * i;
    }
    return ans;
}
int main ()
{
    int i;
    long long n,m;
    double pi=0,pi0=0,x,eps=1e-8;
    scanf("%lf %lld",&x,&n);
    for (i=0; i<n; i++) {
        pi0 = pi;
        pi = pi + pow(-1,i)*pow(x,2*i)/factorial(2*i);
        if(fabs(pi0-pi)<=eps)
        {
            break;
        }
    }
    printf("%.8lf",pi);
    return 0;
}
```

I.撕裂三角形

题目描述

给定三角形三个点的坐标和三角形边上的一个点A的坐标。

求三角形边上的另一个点B使得直线AB平分整个三角形的面积。

输入

第一个数为数据组数n（$1 \leq n \leq 500000$）

接下来n行，每行8个空格分开的整数，前六个数中每两个数表示三角形一个点的坐标，最后两个数表示点A的坐标。所有整数的绝对值≤ 5000。

输出

对于每组数据，输出一行，两个浮点数表示点B的坐标。当你的答案与标准输出的绝对误差小于0.000001时判定你答案正确。

输入样例

```
2
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 1
```

输出样例

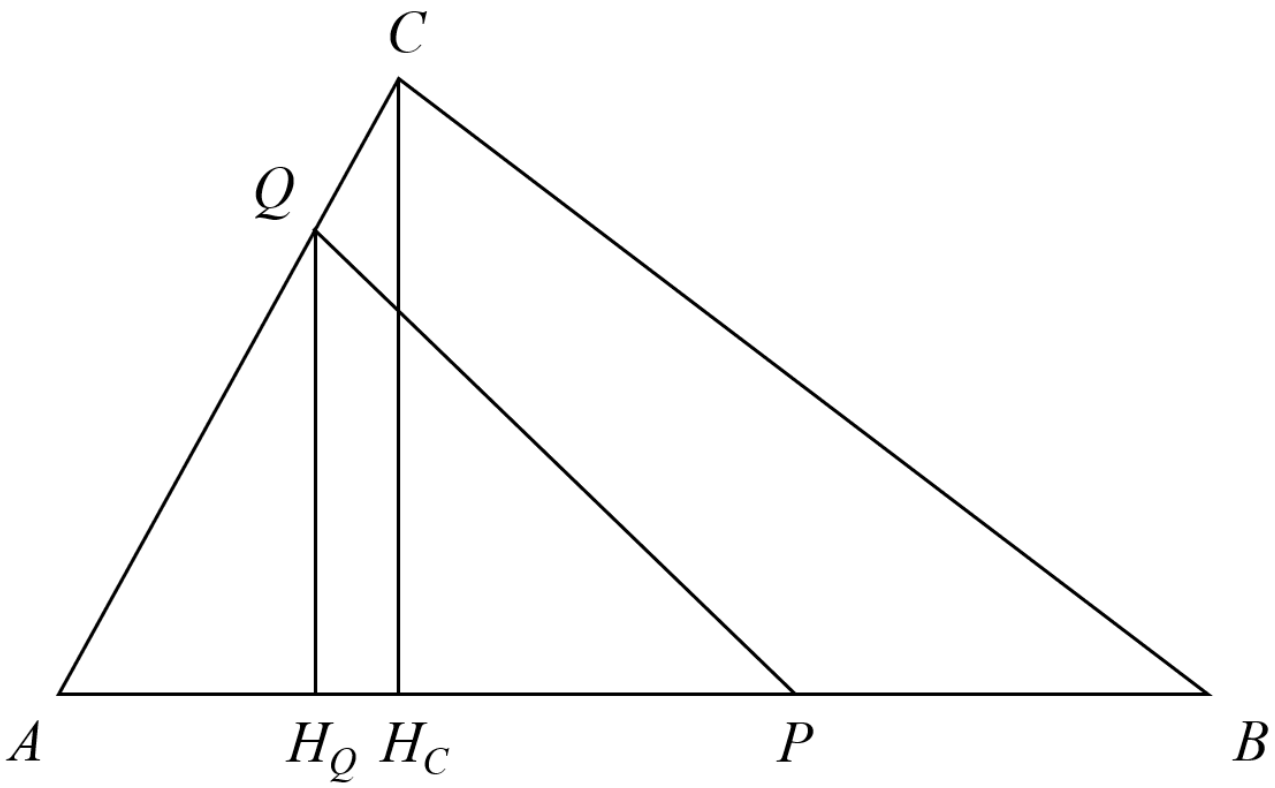
```
0.5 0.5
0.5 0
```

解题思路

本题考察了计算几何基础。

首先，如果给定点是三角形的顶点，答案显然就是对边的中点（中线平分三角形面积）

否则需要进行计算：



- 设所询问点为P，且P不是三角形的任意顶点。
- 记P所在的边为AB，离P近的顶点为B。
- 设答案在Q处，过Q、C分别作AB垂线。
- 由 $S_{\triangle APQ} == \frac{S_{\triangle ABC}}{2}$ 可求出 $|QH_Q|$。
- 由 $|CH_C| : |QH_Q| == |CA| : |QA|$ 可算出 $|QA|$，进而求得 Q 坐标。

按照上述思路进行计算即可，注意叉积的运用。

AC代码

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

#define vint long long
#define SWAP(a, b) ({vint tp=a; a=b; b=tp;})
#define ABS(x) ((x)>=0?(x):-(x))

vint cross(vint ax, vint ay, vint bx, vint by)
{
    return ax*by - ay*bx;
}
vint get_area(vint ax, vint ay, vint bx, vint by, vint cx, vint cy)          // 叉积求面积
{
    vint area = cross(bx-ax, by-ay, cx-ax, cy-ay);
    return ABS(area);
}
double P2P(vint ax, vint ay, vint bx, vint by)
{
    return sqrt((ax-bx) * (ax-bx) + (ay-by) * (ay-by));
}
double P2L(vint px, vint py, vint ax, vint ay, vint bx, vint by)
{

```

```
    return get_area(px, py, ax, ay, bx, by) / P2P(ax, ay, bx, by);}
int is_collinear(vint ax, vint ay, vint bx, vint by, vint cx, vint cy)      // 叉积（面积）为0，三点共线
{
    return !get_area(ax, ay, bx, by, cx, cy);
}

int main()
{
    vint ax, ay, bx, by, cx, cy, px, py;
    double kx, ky;

    int n;
    scanf("%d", &n);

    while (n--)
    {
        scanf("%lld %lld %lld %lld %lld %lld %lld %lld", &ax, &ay, &bx, &by, &cx, &cy, &px, &py);
        if (px == ax && py == ay)
            kx = (bx+cx) / 2.0, ky = (by+cy) / 2.0;
        else if (px == bx && py == by)
            kx = (cx+ax) / 2.0, ky = (cy+ay) / 2.0;
        else if (px == cx && py == cy)
            kx = (ax+bx) / 2.0, ky = (ay+by) / 2.0;
        else
        {
            // 确保P在AB上
            if (is_collinear(px, py, ax, ay, cx, cy))
                SWAP(bx, cx), SWAP(by, cy);
            else if (is_collinear(px, py, bx, by, cx, cy))
                SWAP(ax, cx), SWAP(ay, cy);

            // 确保P离B更近
            if (P2P(ax, ay, px, py) < P2P(bx, by, px, py))
                SWAP(ax, bx), SWAP(ay, by);

            // 计算Q的坐标
            double Q2AB = get_area(ax, ay, bx, by, cx, cy)/2.0 / P2P(px, py, ax, ay);
            double C2AB = P2L(cx, cy, ax, ay, bx, by);
            double ratio = Q2AB / C2AB;
            kx = ax + ratio * (cx-ax);
            ky = ay + ratio * (cy-ay);
        }
        printf("%.10f %.10f\n", kx, ky);
    }

    return 0;
}

```

J.分班帽

解题思路

如果输入的数字 x 是偶数，则四个区域的范围分别是：

红：x：$[1,x/2]$ 、y：$[1,x/2]$

蓝：x：$[1,x/2]$ 、y：$[x/2+1,x]$

绿：x：$[x/2+1,x]$ 、y：$[1,x/2]$

黄：x：$[x/2+1,x]$ 、y：$[x/2+1,x]$

如果输入的数字 x 是奇数，则五个区域的范围分别是：

红：x：$[1,(x-1)/2]$ 、y：$[1,(x-1)/2+1]$

蓝：x：$[1,(x-1)/2+1]$ 、y：$[(x-1)/2+2,x]$

绿：x：$[(x-1)/2+1,x]$ 、y：$[1,(x-1)/2]$

黄：x：$[(x-1)/2+2,x]$ 、y：$[(x-1)/2+1,x]$

中：$x=(x-1)/2+1$ 、$y=(x-1)/2+1$

再根据指定的规则输出对应的字符即可。

AC代码

```
#include <stdio.h>
int main()
{
    int x;
    int i, j;
    char a[] = "Gryffindor", b[] = "SSlytherin", c[] = "RRavenclaw", d[] = "Hufflepuff";
    scanf("%d", &x);
    if(x % 2 == 0)
    {
        for(i = 1; i <= x / 2; i++)
        {
            for(j = 1; j <= x / 2; j++) printf("%c", a[(i + j - 2) % 10]);
            for(; j <= x; j++) printf("%c", b[(i + j - 2) % 10]);
            puts("");
        }
        for(; i <= x; i++)
        {
            for(j = 1; j <= x / 2; j++) printf("%c", c[(i + j - 2) % 10]);
            for(; j <= x; j++) printf("%c", d[(i + j - 2) % 10]);
            puts("");
        }
    }
    else
    {
        for(i = 1; i <= (x - 1) / 2; i++)
        {
            for(j = 1; j <= (x - 1) / 2; j++) printf("%c", a[(i + j - 2) % 10]);
            for(; j <= x; j++) printf("%c", b[(i + j - 2) % 10]);
```

```
            puts("");
        }
        for(j = 1; j <= (x - 1) / 2; j++) printf("%c", a[(i + j - 2) % 10]);
        printf("O");
        for(j++; j <= x; j++) printf("%c", d[(i + j - 2) % 10]);
        puts("");
        for(i++; i <= x; i++)
        {
            for(j = 1; j <= (x + 1) / 2; j++) printf("%c", c[(i + j - 2) % 10]);
            for(; j <= x; j++) printf("%c", d[(i + j - 2) % 10]);
            puts("");
        }
    }
}
```