问题A：能不能更快?

因为想最快，所以尽可能$5m/s$地走
那么完整的$5m/s$要走$x/5$步
如果$x\%5 \neq 0$，那么就说明额外需要一秒，此时要$+1$

```
#include <stdio.h>

int main () {
        int x; scanf("%d", &x);
        int res = x / 5;
        if ( x % 5 ) res ++;
        printf("%d", res);
}
```

问题B：这是我能想到的最难的题了

设置一个flag初始化为0，输入时遍历
如果遇到一个输入为1，则flag变成1
最后看一下如果flag=1说明中间有1，否则中间没有

```
#include <stdio.h>

int main () {
        int n; scanf("%d", &n);
        int flag = 0;
        for ( int i = 0, x; i < n; i ++ ) {
                scanf("%d", &x);
                if ( x == 1 ) flag = 1;
        }
        if ( flag ) printf("HARD");
        else printf("EASY");
}
```

问题 C: 小新同学的整数a+b

设置一个suma和sumb分别代表两个式子的结果
每读入一个int类型就读入一个char类型
如果读入的char类型为'-'那么就是sum-int类型，否则sum+int类型
读入的char类型直到读入进'\n'代表读入结束
计算完之后累加即可

```
#include <stdio.h>

int main () {
        int suma, a; char ca;
        int sumb, b; char cb;

        scanf("%d", &suma);
        while ( scanf("%c", &ca), ca != '\n' ) {
                scanf("%d", &a);
                if ( ca == '+' ) suma += a;
                else suma -= a;
        }
        scanf("%d", &sumb);
        while ( scanf("%c", &cb), cb != '\n' ) {
                scanf("%d", &b);
                if ( cb == '+' ) sumb += b;
                else sumb -= b;
        }
        printf("%d\n", suma + sumb);
}
```

问题 D: B:机智的字符???

设置一个步进的char类型字符，每输出一次后+1，如果到了'z'那么就折回来继续
对于矩阵输出，开两层循环，一个循环表示行数，一个循环表示列数

每一行开始时先输出一个'>'，每一行结束输出一个'<\n'
直到输出m行后结束

```
#include <stdio.h>

int main () {
        char c = 'a';
        int m, n; scanf("%d%d", &m, &n);
        for ( int i = 0; i < m; i ++ ) {
                printf(">");
                for ( int j = 0; j < n; j ++ ) {
                        printf("%c", c);
                        if ( c == 'z' ) c = 'a';
                        else c ++;
                }
                printf("<\n");
        }
}
```

问题 E: 输出最小的正整数

我们设置一个表示最小值的变量minn并初始化为1e18
每次输入一个数进行比较，如果小于minn且是正整数就把minn替换为这个数
最后输出minn

```
#include <stdio.h>

int main () {
        long long minn = 1e18;
        long long n; scanf("%lld", &n);
        for ( long long i = 0, x; i < n; i ++ ) {
                scanf("%lld", &x);
                if ( x < minn && x > 0 ) minn = x;
        }
        if ( minn == 1e18 ) printf("not found");
        else printf("%lld", minn);
}
```

问题 F: 都是素数么

首先素数的判定：如果从2到sqrt(n)没有n的因数，那么n就是素数
这里对这个表达式算一下每个x得到的y值，对于x枚举[a,b]看看是否满足全部的y都是素数

```
#include <stdio.h>

int main () {
        int a, b;
        while ( scanf("%d%d", &a, &b) != EOF, a || b ) {
                int flag = 1;
                for ( int x = a; x <= b; x ++ ) {
                        int y = x * x + x + 41;
                        for ( int j = 2; j * j <= y; j ++ ) {
                                if ( y % j == 0 ) {
                                        flag = 0;
                                }
                        }
                }
                if ( flag ) puts("OK");
                else puts("Sorry");
        }
}
```

问题 G: 一元二次方程

按照高中学的
如果$b^2-4ac<0$那就是没有解
如果$b^2-4ac=0$就是只有一个解：$\frac{-b+sqrt^2-4ac}{2a}$
否则两个解：$\frac{-b+sqrt^2-4ac}{2a}$
输出即可

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    double a, b, c;
    double x1, x2;
    scanf("%lf%lf%lf", &a, &b, &c);
    if(b * b - 4 * a * c < 0) printf("No Answer");
    else if(b * b - 4 * a * c >= 0)
```

```
    {
        x1 = ( - b + sqrt(b * b - 4 * a * c ) ) / 2 / a;
        x2 = ( - b - sqrt(b * b - 4 * a * c ) ) / 2 / a;
        if(x1 == x2) printf("%15.5f", x1);
        else if(x1 != x2)
        {
            if(x1 < x2) printf("%15.5f%15.5f", x1, x2);
            else printf("%15.5f%15.5f", x2, x1);
        }
    }
    return 0;
}
```

问题 H: 制表符

对于每一行输入的字符串
我们在遇到'->'时输出k个空格即可
方法：如果当前位是'-'，下一位没有越界且下一位是'>'，我们输出k个空格，且跳过下一位(下标++)

否则输出原本的字符
每一行结束再加上一个'\n'

```
#include <stdio.h>

int main () {
        int cass;
        for ( scanf("%d", &cass); cass; cass -- ) {
                int n, k; scanf("%d%d", &n, &k);
                while ( n -- ) {
                        char s[10000]; scanf("%s", s);
                        int len = strlen(s);
                        for ( int i = 0; i < len; i ++ ) {
                                if ( s[i] == '-' && i + 1 < len && s[i + 1] == '>' ) {
                                        for ( int j = 0; j < k; j ++ ) printf(" ");
                                        i ++;
                                } else {
                                        printf("%c", s[i]);
                                }
                        }printf("\n");
                }
        }
}
```

问题 I: 小明来签到

一个找规律
首先可以发现，斜着的数总是从左下向右上递进的
斜着的数坐标的和也是相同的，我们设置为id
而且右上总是一个等差数列的和，那么我们可以先利用上面的求出左上三角形内的元素
即固定好id后，$id*(id+1)/2-(i-1)$即可

那么对于右下一半的三角形
我们可以令sum为这一半区域的最小值，就是$n*(n+1)/2+1$
然后我们让$id-(n+1)$求出这是在这一半区域从左往右数第几个斜线上
然后我们算一下这个斜线上最小的数是几，$(n-1+n-id)*id/2$
然后加上需要往上爬的个数$(n-i)$就是答案

```
#include <stdio.h>

int main () {
        long long n, i, j;
        while ( scanf("%lld%lld%lld", &n, &i, &j) != EOF ) {
                if ( i + j - 1 <= n ) {
                        long long id = i + j - 1;
                        long long sum = id * (id + 1) / 2;
                        sum -= i - 1;
                        printf("%lld\n", sum);
                } else {
                        long long sum = n * (n + 1) / 2 + 1;
                        long long id = i + j - 1;
                        id -= n + 1;
                        sum += (n - 1 + n - id) * id / 2;
                        sum += n - i;
                        printf("%lld\n", sum);
                }
        }
}
```

问题 J: 代码格式化

这里是使用C++STL的解法
十分方便，建议学完STL后食用

首先设置缩进数量为ntab
对于每一行输入的字符串设为s，并对这一行的答案字符串设置为res
在遍历时，每次让res加上一个字符s[i]
如果s[i]是一个运算符，那么在前后对res+=" "
如果是';'，说明要换行，那么在res后面放个换行符并缩进一下
若是'{'，说明要再增加一个缩进，同时对最后一个字符之前插入一个空格，然后换行缩进一下
若是'}'，说明前面要少一个缩进，那么就删除一下并且换行，然后对缩进数量-1，锁进一下

```cpp
#include <iostream>
#define ll long long

using namespace std;

int main () {
	string s;
	int ntab = 0;
	while ( getline(cin, s) ) {
		string res;
		for ( int i = 0; i < s.size(); i ++ ) {
			if ( s[i] == '-' || s[i] == '+' || s[i] == '*' || s[i] == '=' || s[i] == '/' || s[i] == '%' ) res += " ";
			res += s[i];
			if ( s[i] == '-' || s[i] == '+' || s[i] == '*' || s[i] == '=' || s[i] == '/' || s[i] == '%' ) res += " ";
			if ( s[i] == ';' ) {
				res += "\n";
				for ( int tab = 0; tab < ntab; tab ++ ) res += "    ";
			} else if ( s[i] == '{' ) {
				res.insert(res.size() - 1, " ");
				res += "\n";
				ntab ++;
				for ( int tab = 0; tab < ntab; tab ++ ) res += "    ";
			} else if ( s[i] == '}' ) {
				res.erase(res.size() - 5, 4);
				res += "\n";
				ntab --;
				for ( int tab = 0; tab < ntab; tab ++ ) res += "    ";
			}
		}
		cout << res << endl;
	}
}
```