注明：以下算法题均用C/C++实现 写在前面：机试题通常有一定的格式，一般分为以下几部分： 1、标题 2、时间和内存的限制(Time Limit, Memory Limit) 3、题目的描述(Description) 4、输入数据的描述(input) 5、输出数据的描述(output) 6、输入数据样例(Sample Input) 7、输出数据样例(Sample Output)

一些注意事项： 1、输出必须和题目一致，不允许多输出任何的内容 2、所有内容的大小写必须一致 3、不能过多或者缺少任何数量的空格和回车符 4、题目输入和输出都是通过标准输入和输出完成，你不需要进行任何文件操作。实际上，出于安全考虑你也不允许进行任何文件操作。

关于输入输出 题目的输入数据不止一项，对于多组数据输入的情况，题目会说明输入数据的结束条件，队员在写处理代码的时候，最简单的方法就是外面一个while (1)的死循环，然后内部对于结束条件进行判断，如果满足条件即break。

有些时候，题目会说以eof表示文件的结束，所以这里给大家介绍一下EOF： EOF的意思是end of file,表示输入的结束。 scanf函数的返回值如果为EOF的话，就表示输入结束了。比如题目要求你求两个数的和，以EOF结束，你就应该这样写：

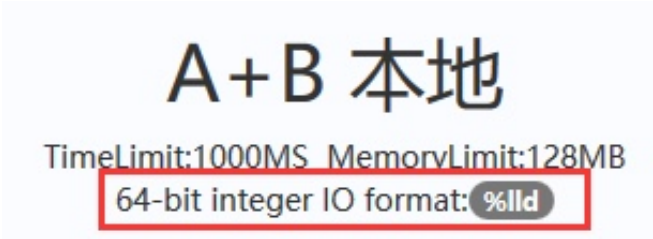
```
while (scanf("%d%d", &a, &b) != EOF) {
    ...
}
```

或者这样：

```
while (1) {
    if (scanf("%d%d", &a, &b) == EOF) {
        break;
    }
}
```

在本机调试程序时，可以通过按 F6 或者 ctrl+z 来输入EOF，一般推荐第一种写法。

另外一些需要注意的事项，以online judge评测为例 Q:64位整型的输入输出格式应该如何表示？ A:每道题目的格式可能是不同的，对应的参数会直接写在题目的标题下，注意查看。



Q:提交后那个返回的结果是什么意思?

A:

Pending... :正在等待评测系统评测

Judging :正在评测，正在等待评测结果

Accepted :恭喜你答案正确，成功解决该题

Wrong Answer :答案错误

Runtime Error :运行错误

Time Limit Exceeded :程序运行时间超出限制

Memory Limit Exceeded :程序运行内存超出限制

Output Limit Exceeded :程序输出的内容过多

Compilation Error :编译错误，请检查代码的语法错误

Presentation Error :格式错误，请检查输出是否多了或者少了 回车空格等字符。

Submit Error :提交错误，遇到此错误联系管理员，可能是因为系统出现BUG导致的

Part I 一些模拟题

一、比较奇偶数个数

第一行输入一个数，为n，第二行输入n个数，这n个数中，如果偶数比奇数多，输出NO，否则输出YES。 **输入描述：**

> 输入有多组数据。 每组输入n，然后输入n个整数（1<=n<=1000）。

输出描述：

> 如果偶数比奇数多，输出NO，否则输出YES。

示例1 输入

```
5 1 5 2 4 3
```

输出

```
YES
```

完整代码实现：

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

```
int main() {
    int i, n, evenCount, oddCount;
    int temp;
    while(scanf("%d", &n) != EOF) {
        evenCount = 0;
        oddCount = 0;
        for(i = 0; i < n;i++) {
            scanf("%d", &temp);
            if(temp % 2 == 0) {
                evenCount++;
            } else {
                oddCount++;
            }
        }
        if(oddCount < evenCount) {
            printf("NO\n");
        } else {
            printf("YES\n");
        }
    }
    return 0;
}
```

二、找最小数

第一行输入一个数n，$1 \leq n \leq 1000$，下面输入n行数据，每一行有两个数，分别是x y。输出一组x y，该组数据是所有数据中x最小，且在x相等的情况下y最小的。

输入描述:

> 输入有多组数据。 每组输入n，然后输入n个整数对。

输出描述:

> 输出最小的整数对。

示例1 输入

```
5
3 3
2 2
5 5
2 1
3 6
```

输出

```
2 1
```

完整代码实现：

```
#include <stdio.h>
#define INT_MAX 0x3f3f3f3f

struct numPair {
    int x, y;
};

int main() {
    int n;
    struct numPair min, temp;
    while(scanf("%d",&n) != EOF) {
        min.x = INT_MAX, min.y = INT_MAX;
        while(n--) {
            scanf("%d %d",&temp.x, &temp.y);
            if(temp.x < min.x || (temp.x == min.x && temp.y < min.y)) {
                min = temp;
            }
        }
        printf("%d %d\n",min.x, min.y);
    }
    return 0;
}
```

三、八进制

> 输入一个整数，将其转换成八进制数输出。

输入描述: 输入包括一个整数$N(0 \leq N \leq 100000)$。

输出描述:

> 可能有多组测试数据，对于每组数据，输出N的八进制表示数。

示例1 输入

> 7 8 9

输出

> 7 10 11

完整代码实现： **递归版本**

```
#include <stdio.h>

void solve(int n) {
    if (n == 0) {
        return;
```

```
    } else {
        solve(n / 8);
        printf("%d", n % 8);
    }
}

int main() {
    int N;
    while (scanf("%d", &N) != EOF) {
        solve(N);
        printf("\n");
    }
    return 0;
}
```

非递归版本

```
#include <stdio.h>
#define MAX_SIZE 1000
void solve(int n) {
    int i = 0, a[MAX_SIZE];
    // 进制转换的过程
    while (n) {
        a[i] = n % 8;
        n /= 8;
        i++;
    }

    i--;
    // 输出
    for(;i >= 0; i--) {
        printf("%d", a[i]);
    }
}

int main() {
    int n;
    while (scanf("%d", &n) != EOF) {
        solve(n);
        printf("\n");
    }
    return 0;
}
```

四、加减乘除

> 根据输入的运算符对输入的整数进行简单的整数运算。 运算符只会是加+、减-、乘*、除/、求余%、阶乘！六个运算符之一。 输出运算的结果，如果出现除数为零，则输出“error”,如果求余运算的第二个运算数为0，也输出“error”。

输入描述:

> 输入为一行。先输入第一个整数，空格输入运算符，然后再空格输入第二个整数，回车结束本次输入。 如果运算符为阶乘！符号，则不输入第二个整数，直接回车结束本次输入。

输出描述:

> 可能有多组测试数据，对于每组数据， 输出一行。输出对输入的两个（或一个）数，根据输入的运算符计算的结果，或者“error”。

示例1 输入

> 12 + 34 54 - 25 3 * 6 45 / 0 5 ! 34 % 0

输出

> 46 29 18 error 120 error

完整代码实现：

```
#include <stdio.h>
void solve(int a, char op) {
    int i, b, count;
    if(op == '!') {
        count = 1;
        for(i = a;i >= 1;i--) {
            count = count * i;
        }
        printf("%d\n",count);
    } else {
        scanf("%d",&b);
        if(op =='+') {
            printf("%d\n",a + b);
        }
        if(op == '-') {
            printf("%d\n",a - b);
        }
        if(op == '*') {
            printf("%d\n",a * b);
        }
        if(op == '/') {
            if(b != 0) {
                printf("%d\n",a / b);
            } else {
                printf("error\n");
            }
        }
        if(op == '%') {
            if(b != 0) {
                printf("%d\n",a % b);
            } else {
                printf("error\n");
```

```
            }
        }
    }
}
int main() {
    int a;
    char op;
    while(scanf("%d %c",&a,&op) != EOF) {
        solve(a, op);
    }
    return 0;
}
```

五、大数加法

> 所谓的高精度运算,是指参与运算的数(加数,减数,因子……）范围大大超出了标准数据类型（整型，实型）能表示的范围的运算。例如，求两个200位的数的和。这时，就要用到高精度算法了。高精度运算主要需要解决的问题： 1、加数、减数、运算结果的输入和存储: 用字符串输入，用数组来存储 2、运算过程 ： 一位一位模拟人列竖式的方式来计算 3、结果的输出 ： 一位一位输出 题目要求： 请你用高精度算法求两个非负数的和，这两个数的最长位数为400位。

输入描述:

> 输入有多组测试数据，每组数据占一行，分别表示加数a和被加数b，a,b间用空格隔开。a, b均为非负数。

输出描述:

> 输出对应每行输入，输出对应的a,b值的和。

示例1 输入

> 99999999999999999999999999 99999999999999999999999999999999 99999999999999999999999999
> 1 99999999999999999999999999 88888888888888888888888888888888

输出

> 100000099999999999999999999999998 10000000000000000000000000000
> 88888898888888888888888888888887

完整代码实现：

```
#include <stdio.h>
#include <string.h>
#define MAX(a, b) (a > b ? a : b)
#define MAX_N 410
```

```c
void solve(char *str1, char *str2) {
    int ans[MAX_N] = {0};
    int i, j, len1, len2, maxLen, tempMaxLen;

    //临时变量设置初始值
    len1 = strlen(str1);
    len2 = strlen(str2);
    i = len1 - 1;
    j = len2 - 1;
    maxLen = MAX(len1, len2);
    tempMaxLen = maxLen;

    //模拟竖式运算，从右往左的运算顺序，空出第0位不用，以作为最高位的进位位
    while (i >= 0 && j >= 0) {
        ans[tempMaxLen] = str1[i] + str2[j] - 2 * '0';  //将相加后的结果转化成整型
        tempMaxLen--;
        i--;
        j--;
    }

    //将剩余部分相加
    for (i = 0; i < len1 - len2; i++) {
        ans[i + 1] = str1[i] - '0';
    }
    for (i = 0; i < len2 - len1; i++) {
        ans[i + 1] = str2[i] - '0';
    }

    //处理进位
    for (i = maxLen; i > 0; i--) {
        if (ans[i] >= 10) {             //说明该位相加后的结果需要进位
            ans[i] -= 10;               //“该位” - 10
            ans[i - 1] += 1;            // 向前一位进1
        }
    }

    //看最高位的进位位是否为1
    //为1则说明有低位向最高位进位,则需从第0位开始输出
    //则不为1则没有,从第1位开始输出即可
    i = ans[0] == 1 ? 0 : 1;
    for (; i <= maxLen; i++) {
        printf("%d", ans[i]);
    }
    printf("\n");
}

int main() {
    char str1[MAX_N], str2[MAX_N];
    while(scanf("%s %s", str1, str2) != EOF) {
        solve(str1, str2);
    }
```

```
}
```

类似题目：

> 题目要求：请你用高精度算法求两个非负数的差，这两个数的最长位数为400位。输入数据保证差为非负数。即：第一个数总是不小于第二个数。

输入描述：

> 有多组测试数据，每组数据占一行，分别表示被减数a和减数b，a,b间用空格格开。a,b均为非负数。 输入数据保证差为非负数。即：第一个数总是不小于第二个数。

输出描述：

> 输出对应每行输入，输出对应的a,b值的差。

示例1 输入

> 9 9 990 90 1900 900 1900 1000 99999999999999999999999999 1

输出

> 0 900 1000 900 99999999999999999999999998

完整代码实现：

```c
#include <stdio.h>
#include <string.h>
#define MAX_N 410

void solve(char *str1,char *str2) {

    int ded[MAX_N] = {0};
    int i, j, len1, len2;

    //临时变量设置初始值
    len1 = strlen(str1);
    len2 = strlen(str2);
    i = len1 - 1;
    j = len2 - 1;

    //模拟竖式运算，从右往左的运算顺序
    while (i >= 0 && j >= 0) {
        ded[i] = str1[i] - str2[j];  //将相减后的结果转化成整型
        i--;
        j--;
    }

    //剩余部分
    while(i >= 0){
```

```
        ded[i] = str1[i] - '0';
        i--;
    }

    //处理借位
    for (i = len1 - 1; i > 0; i--) {
        if (ded[i] < 0) {
            ded[i] += 10;
            ded[i - 1] -= 1;
        }
    }

    //寻找第一个不为0的位
    for(i = 0; i < len1 - 1; i++) {
        if(ded[i] != 0) {
            break;
        }
    }

    for (; i <= len1 - 1; i++) {
        printf("%d", ded[i]);
    }
    printf("\n");
}
int main(){
    char str1[MAX_N], str2[MAX_N];
    while (scanf("%s %s", str1, str2) != EOF) {
        solve(str1,str2);
    }
    return 0;
}
```