E3-Statement

- 练习时间2022/03/26 08:00 - 2022/03/30 00:00
- 总共11道题，编号为A~K，**题目不一定按难度排序**，同学们可以按照通过人数答题；或者提前读完所有题目，按照自己擅长题目的情况答题。
- 本次比赛的最后三道题目是C4的预习题，预习题计1分。
- 各题分值分布为如下，总分100+3分：

序号	A	B	C	D	E	F	G	H	I	J	K
分值	20	20	20	15	10	5	5	5	1	1	1

- 请严肃练习，严禁抄袭他人代码，课程组会在练习结束后进行代码查重并给予警告。

组题：爱吃猪脚的猪脚

更多题目作者：cbd的学生cxccxc、ljh、wqh、Arthas、lzq、wwh、魔法少女zhn、呱呱泡蛙、已经毕业的前助教头子dch

---

A 数数逆序对

题目描述

cxccxc在上《线性代数》这门课时被行列式的计算方法难住了。

计算行列式涉及到计算一个排列的逆序对数，由于cxccxc不太会数数，导致逆序对数总是数错，你能编写程序帮帮他吗

在一个数列 $a_1, a_2, a_3, \ldots, a_n$中，若存在 $1 \le i < j \le n$，满足 $a_i > a_j$，则称 (a_i, a_j) 为一个逆序对。现在给出一个数列，请你计算逆序对的个数

输入格式

输入共两行

第一行只有一个正整数 n ，表示这个数列的元素个数

第二行包含 n 个由空格隔开的整数 a_i ，为这个数列的每一个元素

输出格式

输出一个整数，为这个排列中的逆序对数

输入样例

```
6
1 6 4 3 5 2
```

输出样例

```
8
```

样例解释

所有的 8 个逆序对为 $(6,4),(6,3),(4,3),(6,5),(6,2),(4,2),(3,2),(5,2)$

数据范围

$0 < n \le 1000$

$1 \le a_i \le 1000$

Author: cbd的学生cxccxc

B 完全数!

题目描述

小林今天在对 28 分解质因数时发现了一个奇怪的现象：

28 的所有真因子（除自身外的约数）为：$1,2,4,7,14$，而将这些因子全部相加恰好等于 28。

查阅资料发现，一个数符合所有真因子（除自身外的约数）之和等于本身，则这个数便是完全数又称完备数、完美数。完全数有许多神奇的特性，感兴趣的同学可以自行百度。

现在，小林想知道对于给定的数 x_i，它是否是一个完全数，你能帮帮他吗？

输入

第一行，一个正整数 n，表示需要判断的数的总数；

接下来 n 行，每行一个正整数数 x_i，表示需要判断的数。

输出

共 n 行，第i行输出x_i是否为完全数。若是，则输出 `YES`，否则输出 `NO`。

输入样例

```
4
6
8
10
28
```

输出样例

```
YES
NO
NO
YES
```

数据范围

$1 \le n \le 100$，$1 \le x_i \le 10000$

AUTHOR：ljh

C 求e的近似值

题目描述

小林今天学习了复变函数，其中号称“最完美的数学公式”的欧拉公式 $e^{i\pi}+1=0$ 吸引了小林的注意，因为它包含了自然常数 e 在内的五个最重要的数学元素。

众所周知，自然常数e可以用级数 $1+\frac{1}{1!}+\frac{1}{2!}+\cdots+\frac{1}{n!}$ 来近似计算，现在小林想知道在给定 n 的值的情况下，e的近似值是多少。

小林当然能很快地给出答案，他只是单纯地想考考你。

输入

一个正整数 n。

输出

一个浮点数，即此时 e 的近似值，结果保留小数点后 14 位。

输入样例

```
3
```

输出样例

```
2.66666666666667
```

数据范围

$1 \le n \le 10^9$。

HINT

老老实实地模拟迭代是会超时的！！！

思考一下迭代多少次后就可以不用迭代了。

AUTHOR：ljh

D 统计胜场数

题目描述

n 支球队参加比赛，**两两对决**，每支参赛队伍只能与其他队伍相遇一次，请你帮忙统计每支球队的胜场数。

输入格式

第一行一个正整数 n （$2 \le n \le 5$），表示参加比赛的球队数；

接下来若干行，每行格式为 `%c %c %c` ，分别对应表示 $c1$、$result$、$c2$，其中 $c1$、$c2$ 表示球队名，用从 `A` 开始的大写字母表示，$result$ 表示比赛结果，非输即赢，用 `>` 或 `<` 符号表示，$result$ 为 `>` 时表示球队 $c1$ 战胜 $c2$ ， `<` 表示球队 $c2$ 战胜 $c1$。

输出格式

输出共 n 行，每行格式为 `%c:%d` 表示 `球队名:胜场数`；

输出按照**胜场数由多到少**排列，若胜场数相同，则按照**球队名字母表顺序**排列。

输入样例

```
4
A > D
B > A
B < C
C > A
D < C
D > B
```

输出样例

```
C:3
A:1
B:1
D:1
```

HINT

- 清除行末换行符，可参考 `C1-I` 中的处理方式，也可使用课件上的 `while (getchar() != '\n');` 但由于最后一行可能没有换行符，因此可能会导致**死循环**，使用这种方法更安全的写法是：

  ```
  c = getchar();                    // 尝试读取一个字符
  while (c != '\n' && c != EOF)    // 若该字符不是换行符也不是EOF
      c = getchar();                // 再继续读取
  ```

- 根据题意，每支球队最多赢 4 场比赛，因此按字母表顺序枚举每一支球队，看是否有赢 4 场、3 场、...、0 场比赛的球队，有就直接输出，即能达到排序的效果。

Author: wqh

E 直线数局

题目描述

有一个由整数组成的长链，如图所示。

3	5	1	-2	5	3	-1	-2	2	-4	1	4

每一个方格上的数字代表到达它时下一步行走的规则，例如5表示下一步需要向右走5格，-2表示下一步需要向左走2格。

现在，给你一个直线数局，从最左侧的方格开始走，请判断出它是否能走到最右侧的方格。

输入

第一个数为数局长度n

接下来一行，为n个由空格隔开的整数，表示从左到右每一格的数字。

输出

输出两行。

第一行，若能走到最后一格，输出`True`，否则输出`False`。

第二行，若能走到最后一格，输出行走的步数，若不能，输出到达最远格的序号（最左边为第`0`格）。

输入样例1

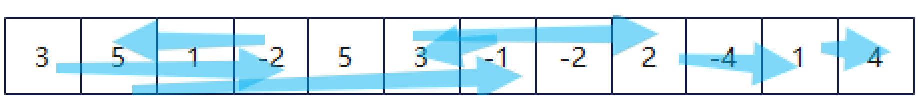
```
12
3 5 1 -2 5 3 -1 -2 2 -4 1 4
```

输出样例1

```
True
7
```

样例解释1

路线如图所示



可到达最右侧方格

输入样例2

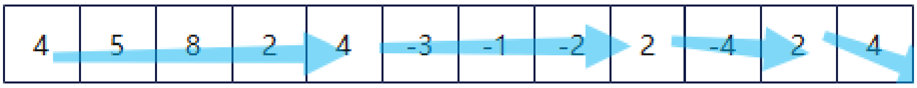
```
12
4 5 8 2 4 -3 -1 -2 2 -4 2 4
```

输出样例2

```
False
10
```

样例解释2

路线如图所示



路线直接越出了数字链，因此无法到达。

数据范围

$n < 100$

方格中的数在`int`范围内（若为0，则代表到达后只能永远停留在该方格，成为死局，故无法到达）

HINT

1. 仔细看数据范围中的第二条，挖掘重要信息。
2. 为什么这个题的题名读着很别扭？因为进阶版读着就不别扭了啊。

Author: Arthas

F 溢0数

题目描述

考虑一系列整数的乘积，其在二进制下末尾有几个0

输入格式

多组数据输入

每组两行，第一行，一个整数`n`，表示接下来输入数的个数

第二行，`n`个`int`范围内的十进制**非零**整数

输出格式

每组数据一行，一个整数，表示这些十进制数相乘后结果在二进制下末尾的`0`的个数

输入样例

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

输出样例

```
1
1
3
```

样例说明

第一组，乘积为 2，其二进制表示为 10 ，输出 1

第二组，乘积为 6 ，其二进制表示为 110 ，输出 1

第三组，乘积为 24 ，其二进制表示为 11000 ，输出 3

数据范围

$1 \leq n \leq 1000$

##HINT

数在计算机中以补码的形式储存

彩蛋

以下内容对解题没有帮助，只是作为出题感想

这道题用`python`生成数据，也可以说溢0数全靠`py`

G 高精度加法

题目描述

计算A+B

输入

第一个数为数据组数 n（$n \leq 10$）

接下来n行，每行2个整数A，B（ 其中 $0 \leq A, B \leq 10^{500}$）

输出

对于每组数据，输出一行，代表 A+B 的值

输入样例

```
3
1 2
7821837687216312872167367 8712637821637861278
999999999999999999999999 999999999999999999999999999
```

输出样例

```
3
7821924813594529250780145
1000999999999999999999999998
```

##hint
long long的数据范围为 -9223372036854775808 ~ +9223372036854775807 是不足以满足本题计算要求的。需要大家使用数组模拟竖式运算。

还需要注意的点

1.由于数字很长，输入时请使用字符串输入

```
char a[510];
scanf("%s",a);
```

2.可以使用头文件`string.h`中的`strlen`函数计算字符串长度，例如：

```
int length=strlen(a);
```

3.最高位可能会有进位

AUTHOR：wwh

H zhn の money

题目描述

zhn本学期课**很少**（真的很少），他通过打工获得了一些 money，一共有 m 元，于是他决定请 xf 去吃 $crazy\ Thursday$。但是由于 zhn 很懒，他想把自己有的 m 元装到 n 个钱袋里面，以便交钱时候方便，具体装配规则如下：

- **不能**有两个钱袋装有相同的大于 1 的钱数。
- 在他**钱的总数**的支付能力下，任何数目的 Money 他都能用这些封闭好的小钱的组合来付账。

  (也就是说，他能通过这些钱袋表示从 1 到 m 的**所有数字**)
- 总钱袋数量**最少**

请你输出他所需要的钱袋数量。

输入

输入一行，一个单个整数 m（$1 \le m \le 10^9$），表示 zhn 本学期赚到的钱的总数

输出

输出一个数 n，钱袋的最少数量。

输入样例

```
3
```

输出样例

```
2
```

Hint

进制，打表找规律

author： 魔法少女zhn

后记:

*“ xf这个名字可真难写，倒不是笔画繁琐，只是写的时候要蘸上四分黄昏，三分月色，两分微醺，还有一分她的可爱。” *

I 林士谔算法

本题为C4预习题

题目介绍

一个神奇的事情是，隔壁信息类（士谔书院）的学生没有学过林士谔算法。今天在这里介绍一下林士谔先生提出的算法。

林士谔（1913-1987）是北航建校元老之一，中国自动控制专家、航空教育家，1939 年获美麻省理工学院（MIT）博士学位。在他的博士论文《飞机自动控制理论》中，林士谔创造性地首次提出了劈因法逐步求解高次实系数多项式的根。在 1940 年 8 月、1943 年 8 月和 1947 年 7 月，林士谔先后在 MIT 出版的《数学物理》杂志上接连正式发表了3篇关于解算高阶方程式复根方法的论文，每次均有改进，获得了当时国际数学界相当高的评价。

在当时计算机科学尚不发达的情况下，要解四阶以上的高阶代数方程，非常困难，林士谔求实系数代数方程的复根时，通过不断迭代修正，直至达到要求精度的办法，**找出一个二次因子**来求得方程的复根。这种求实系数代数方程的复根方法，避免了复数运算。[1]

上世纪四、五十年代，只能以简单的工具（如手摇计算器），用人工的方式按照林氏法求解多项式的根，费时、费力，多项式次数和精度都受到限制。广泛应用数字计算机后，输入多项式的系数，很快就能输出结果，上述缺陷不再存在，这是计算机程序包（如MATLAB）中的多项式求根程序显示的威力，而编制这种程序依据的原理正是林氏劈因法，充分表明了在当今计算机时代林氏法的巨大生命力。这个以林士谔命名的方法至今还在被发展，并应用现代计算机来进行快速运算。

[[1]林士谔. 论劈因法解高阶特征方程根值的应用问题[J]. 数学进展, 1963(03):207-217.]

——以上为背景资料——

本题中的算法可以对一个次数大于等于 3 的多项式 $f(x) = a_n x^n + ... + a_1 x + a_0$ 进行因式分解，求出一个形如 $x^2 + px + q$ 的二次因子，请在以下程序的基础上补充输入和输出代码，将输入的多项式的该因子输出。

```
//全局变量
double b[20]; //b是多项式a除以当前迭代二次三项式的商
double c[20]; //c是多项式b乘以x平方再除以当前迭代二次三项式的商
double p;     //p是待求的一次项
double q;     //q是待求的常数项

void Shie(double a[], int n)//a是原始的多项式，n是多项式次数
{
    memset(b,0,sizeof(b));
    memset(c,0,sizeof(c));
        p = 0;
    q = 0;
    double dp = 1;
    double dq = 1;
    while (dp > 1e-12 || dp < -1e-12 || dq > 1e-12 || dq < -1e-12)
    {
        double p0 = p;
        double q0 = q;
        b[n - 2] = a[n];
        c[n - 2] = b[n - 2];
        b[n - 3] = a[n - 1] - p0 * b[n - 2];
        c[n - 3] = b[n - 3] - p0 * b[n - 2];
        int j;
        for (j = n - 4; j >= 0; j--)
        {
            b[j] = a[j + 2] - p0 * b[j + 1] - q0 * b[j + 2];
            c[j] = b[j] - p0 * c[j + 1] - q0 * c[j + 2];
        }
        double r = a[1] - p0 * b[0] - q0 * b[1];
        double s = a[0] - q0 * b[0];
        double rp = c[1];
        double sp = b[0] - q0 * c[2];
        double rq = c[0];
        double sq = -q0 * c[1];
        dp = (rp * s - r * sp) / (rp * sq - rq * sp);
        dq = (r * sq - rq * s) / (rp * sq - rq * sp);
        p += dp;
        q += dq;
    }
}
```

其中，n 是多项式次数，至少为 3，需要将待分解多项式**对应次数的系数**存入a数组**对应下标**当中。

算法的原理比较复杂，不做介绍。

输入格式

多组数据读入。每一组：

第一行是给定多项式次数 n，至少是 3 次。多项式次数不超过 10 次。

第二行是**降幂排列**的给定多项式，各项系数均为浮点数，保证首项系数不为 0。如果缺项，在对应位置会给出 0。

输出格式

对每一组数据，输出一行：

输出给定多项式的指定二次三项式因子，要求首项系数为 1，如果缺项，对应位置是 0。各项系数都要保留 6 位小数。

无需修改算法内容中的初值。如果初值改动，可能会查找到其他因子。

输入样例

```
6
1.0 -4.0 11.0 -18.0 22.0 -16.0 8.0
10
1 0 -1 0 -1 0 -1 0 1 0 1
```

输出样例

```
1.000000 -1.000000 2.000000
1.000000 0.000000 0.618034
```

样例解释

多项式$x^6-4x^5+11x^4-18x^3+22x^2-16x+8$含有因子$x^2-x+2$。

HINT

本题考察此段代码的正确调用方式。你可以在本地跑一跑，猜一猜怎样调用执行这段代码。

这里的板子写成了全局变量。更好的办法，等到学过指针之后，把p和q按照传址的办法传进去。

AUTHOR：呱呱泡蛙

搬运与整理：爱吃猪脚的猪脚

J 测评机没有 sqrt()！

本题为C4预习题

题目描述

计算一下开方吧！

连任了若干年C语言的助教“爱吃猪脚的猪脚”把测评机上的 `sqrt()` 函数删掉了，所以你不能用 `sqrt()` 。

聪慧的呱呱泡蛙想了想，顺手把“pow”、“exp”和“log”也一并禁掉了。

呱呱泡蛙，发生甚么事了？

该`spj`的原理是，提交代码中不得出现上述连续的字母序列，出现即判 Other Error。

输入

共一行。

一个实数 a ，且 $0 \le a \le 1000$ ，请计算 $\sqrt{a}$ 。

输出

共一行。

一个实数，表示 $\sqrt{a}$ ，保留 8 位小数。

输入样例

```
12.3
```

输出样例

```
3.50713558
```

Hint 1

试试二分法或者牛顿法吧。牛顿法请自行学习，人教版高中数学课本上就有，很简单。

Hint 2

一种神奇的办法也可以参考萌娘百科数字梗页面的"算法竞赛"代码段。（萌娘百科）其中有这样一段代码。输入`float`类型的32位浮点数 x ，将 x 转换为 $\sqrt{x}$ 。

```
float quickinvroot(float x){
    float xhalf = 0.5f * x; //32位浮点数
    int i = *(int*)&x; //按位强制按32位整数理解
    i = 0x5f375a86 - (i >> 1); //整数位运算
    x = *(float*)&i; //按位强制按32位浮点数理解
    x = x*(1.5f-(xhalf*x*x)); //再来一次牛顿法
    return 1.0f/x; // 返回x的开方需要计算倒数
}
```

它的解释可以参考这个链接。（cnblogs）或者知乎。（知乎）

一个`float`正浮点数有8位指数 E 和23位小数 M 。浮点数的值为：

$$F = 2^{E-127}(1 + M \times 2^{-23})$$

强制按32位整数理解的值为：

$$I = E \times 2^{23} + M$$

对两者进行一番变形，有：

$$127 + \log_2 F = E + \log_2(1 + M \times 2^{-23})$$

$$I \times 2^{-23} = E + M \times 2^{-23}$$

在 0 到 1 区间之内，$\log_2(1+x)$ 可以用线性函数 $x+k$ 近似， k 为待定的小参数。这样近似的结果是，下面的等式**近似成立**（并非完全成立，只是近似）：

$$\log_2 F = I \times 2^{-23} - 127 + k$$

于是对于 $F_y = \frac{1}{\sqrt{F_x}}$ 的关系有 $\log_2 F_y = -\frac{1}{2}\log_2 F_x$ ，即：

$$I_y \times 2^{-23} - 127 + k = -\frac{1}{2}(I_x \times 2^{-23} - 127 + k)$$

$$I_y = \frac{3}{2}(127 - k) \times 2^{23} - \frac{1}{2}I_x$$

这就是语句`i = 0x5f375a86 - (i >> 1);`的含义。

因为`float`表示精度不到8位小数，请使用64位的`double`和`long long`代替，64位的`double`含有11个指数位和52个小数位。并且使用数字`0x5FE6EB50C0000000LL`（这数是呱呱泡蛙算出来的），以及至少3次牛顿法。注意0的位置的特判。

去年的spj游戏是助教小智酱玩的吧？

Author: 爱吃猪脚的猪脚 && 呱呱泡蛙

K 荷家军 进攻 汉诺塔

本题为C4预习题

题目描述

汉诺塔是一个经典的递归问题，为了赚取理财的本金，荷家兄弟姐妹决定将搬动汉诺塔来获得压在最大的盘下面的小钱钱。

现在有一个三柱汉诺塔，这三个柱子各有标记（一个大写英文字母）。在左边的柱上有`n`个圆盘，从上到下，每个圆盘依次被编号为：1,2,3,...,n。请你输出将该柱上的盘子全部搬动到最右边的柱子的最快方法。

搬动过程中规则如下：

1. 一次只能搬动一片圆盘
2. 编号大的圆盘不能出现在编号小的圆盘上面

输入格式

一行，一个正整数`n`（$1 \le n \le 18$）和三个大写英文字母`x`,`y`,`z`（依次表示左、中、右三个柱的标记，不会重复），输入内容用空格分开。

输出格式

每一步输出一行，其格式为`move {圆盘编号} from {从哪个柱来（柱标号）} to {移到哪个柱（柱标记）}`

输入样例

```
3 L M R
```

输出样例

```
move 1 from L to R
move 2 from L to M
move 1 from R to M
move 3 from L to R
move 1 from M to L
move 2 from M to R
move 1 from L to R
```

AUTHOR:dch