答题前提示：

注意样例的**输入输出格式要求**，什么时候换行，什么时候空格，有几个空格？

注意题目是否提供**输入边界情况**

题目难度不完全递增，一时间没思路可以先往后看看

（如果时间充足不妨用多种算法解决题目，优化算法来降低时间复杂度（主要）和空间复杂度）

1、约瑟夫环

问题来历

编辑

据说著名犹太历史学家Josephus有过以下的故事：在罗马人占领乔塔帕特后，39 个犹太人与Josephus及他的朋友躲到一个洞中，39个犹太人决定宁愿死也不要被敌人抓到，于是决定了一个自杀方式，41个人排成一个圆圈，由第1个人开始报数，每报数到第3人该人就必须自杀，然后再由下一个重新报数，直到所有人都自杀身亡为止。然而Josephus 和他的朋友并不想遵从。首先从一个人开始，越过k-2个人（因为第一个人已经被越过），并杀掉第*k*个人。接着，再越过k-1个人，并杀掉第*k*个人。这个过程沿着圆圈一直进行，直到最终只剩下一个人留下，这个人就可以继续活着。问题是，给定了和，一开始要站在什么地方才能避免被处决。Josephus要他的朋友先假装遵从，他将朋友与自己安排在第16个与第31个位置，于是逃过了这场死亡游戏。[1]

现在将 n 个人围成一个圈，分别为他们编上序号，从第一个人开始报数，每当报数到 3 的人退出游戏，然后再由下一个人重新报数，直到剩下最后一个人。

要求打印出每次退出游戏的人的序号，以及找出最后留在圈子中的人原来的序号。

示例:

输入:

13

输出:

3 6 9 12 2 7 11 4 10 5 1 8

13

2、结构体加指针

树木有很多种类，也有它自己的树龄，这都属于它的独有元素

(1)现建立一个结构体 tree，成员变量：种类 type(字符串)、树龄 age(整型)

(2)构造一个函数 Input 用以对 tree 的成员变量进行赋值

(3)实现一个空函数(返回值为空) swap 来交换两个 tree 的成员变量

age

tips：swap 形参为两个指针

3、脑筋急转弯

如何判断两个整数是否相等呢？6 和 110 会相等吗？
当 6 是一个十进制数，110 是一个二进制时，6=110。
所以，现在请判断题目给定的两个整数是否相等，若相等，输出二者的进制

（最小进制）

！ 题目保证两个整数在二进制到十进制之间

示例:

输入 1:

6 110

输出 1:

Yes 7 2

输入 2:

1 10

输出 2:

No

4、寻找最简分数

给你一个整数 n ，请你输出所有 0 到 1 之间（不包括 0 和 1）满足分母小于等于 n 的 最简 分数 。分数可以 任意 顺序输出。

若不存在这样的分数则输出”error”

示例 ：

输入 1：

2

输出 1：

1/2

输入 2：

4

输出 2：

1/2 1/3 1/4 2/3 3/4

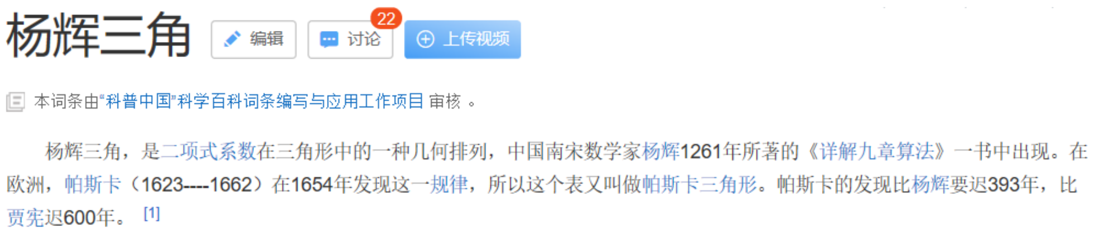
输入 3：

1

输出 3：

error

5、杨辉三角



杨辉三角

编辑 讨论 22 上传视频

本词条由"科普中国"科学百科词条编写与应用工作项目 审核 。

杨辉三角，是二项式系数在三角形中的一种几何排列，中国南宋数学家杨辉1261年所著的《详解九章算法》一书中出现。在欧洲，帕斯卡（1623----1662）在1654年发现这一规律，所以这个表又叫做帕斯卡三角形。帕斯卡的发现比杨辉要迟393年，比贾宪迟600年。 [1]

以下是杨辉三角的一些基础特征

前提：每行端点与结尾的数为1.

（与上图中的n不同，这里第一行定义为n=1）

1. 每个数等于它上方两数之和。
2. 每行数字左右对称，由1开始逐渐变大。
3. 第n行的数字有n项。
4. 前n行共[(1+n)n]/2 个数。

对于给定的 n，输出第 n 行的全部数据
(1) 递归算法实现
(2) 非递归算法实现

示例:

输入:

6

输出:

1 5 10 10 5 1

6、吃桃子

现总共有 n 个桃子，你决定每一天选择如下方式之一吃这些桃子：

吃掉一个桃子。
如果剩余桃子数 n 能被 2 整除，那么你可以吃掉 n/2 个桃子。
如果剩余桃子数 n 能被 3 整除，那么你可以吃掉 2*(n/3) 个桃子。

每天你只能从以上 3 种方案中选择一种方案。
请你计算吃掉所有 n 个桃子的最少天数。

示例:

输入:

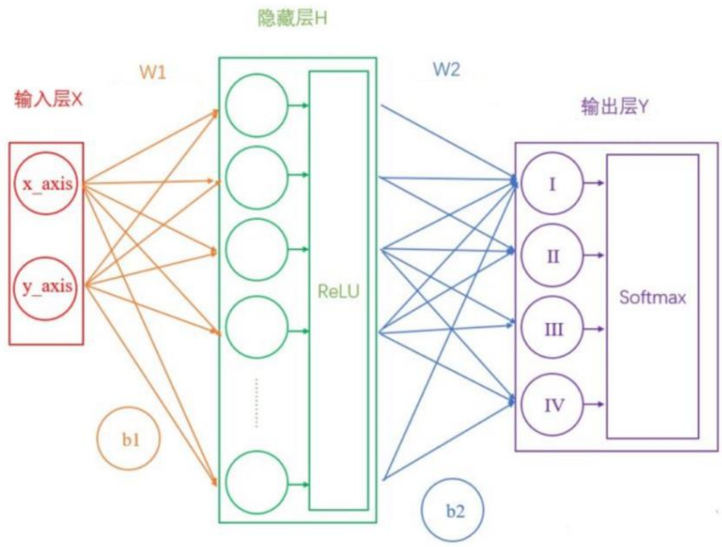
10

输出:

4

/*
第一天可以选择吃掉 1 个桃子（剩 9 个），也可以吃掉 5 个桃子（剩 5 个），以此类推。
10 个桃子的最短时间为 1-6-2-1 四天
*/

7、函数处理



图片看起来有点复杂，但我们要做的只是简化版的函数处理

(1) 输入层

给出的输入数据 x

(2) 从输入层到隐藏层

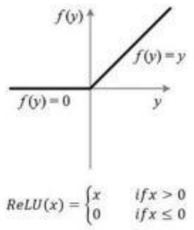
连接输入层和隐藏层的是 W1 和 b1。

$$H=X*W1+b1$$

(3) 激活层

数据在经过激活层时用激活函数进行了处理

我们这里采用 ReLU 激活函数



(4) 从隐藏层到输出层

连接隐藏层和输出层的是 W2 和 b2

$$Y=H*W2+b2$$

(5) 输出层（**Softmax** 层）

我们想让最终的输出为概率

计算公式：

$$S_i = \frac{e^i}{\sum_j e^j}$$

现要对给出的一个数组 X 进行处理。X、W1、b1、W2、b2 由输入给出，请输出 S_i

note：

1、#define e 2.71828

2、#include <math.h>

示例 ：

输入 ：(//后为注释)

5 //数组容量

1 6 7 2 3 //数组里的数据

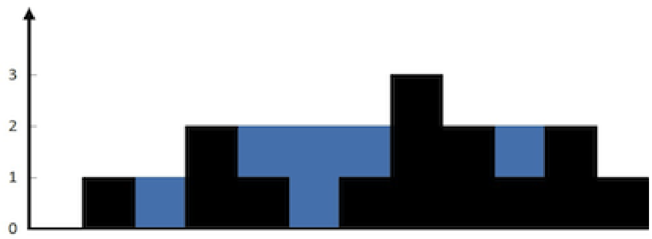
3 -2 0.5 4 //W1 b1 W2 b2

输出 ：

```
5
1 6 7 2 3
3 -2 0.5 4
0.000100638 0.181956 0.815471 0.000451026 0.00202136
```

8、接收液体

将物块以一定的秩序堆积起来，可以形成一个接收液体的容器。
现给定 **n** 个非负整数表示每个宽度为 1 的物块的堆积情况，请计算出其能接受液体的数量（具体看案例）



上面是由数组［0，1，0，2，1，0，1，3，2，1，2，1］表示的堆积情况，在这种情况下，可以接 6 个单位的液体（蓝色部分表示液体）。

示例:

输入:

12

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

输出:

6

9、行列式的计算

在学习线性代数时，行列式的计算是不可缺少的一环，现需要对给定的行列式进行计算并输出他的值
以下是行列式的计算方法:

n阶行列式的计算

首先给出代数余子式的定义。

定义2 [1] 在行列式

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$$

中划去元素a_{ij}所在的第i行第j列，剩下的$(n-1)^2$个元素按原来的排法构成一个n-1阶的行列式M_{ij}，称M_{ij}为元素a_{ij}的余子式，$A_{ij}=(-1)^{i+j}M_{ij}$称为元素的代数余子式。

定理 [1] 设

$$d = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix},$$

A_{ij}表示元素a_{ij}的代数余子式，则下列公式成立：

$$a_{k1}A_{i1} + a_{k2}A_{i2} + \cdots + a_{kn}A_{in} = \begin{cases} d, k=i, \\ 0, k\neq i. \end{cases}$$

$$a_{1l}A_{1j} + a_{2l}A_{2j} + \cdots + a_{nl}A_{nj} = \begin{cases} d, l=j, \\ 0, l\neq j. \end{cases}$$

或

消元法。
行列式某一行 × 某个数 加到另外一行， 行列式的值不变。

利用这个把行列式变成

a11 0 0 0 0
a21 a22 0 0 0
a31 a32 a33 0 0
a41 a42 a43 a44 0
....
an1 an2 an3 an4 ...... ann

的形式

则行列式的值等于 a11*a22*a33*a44*......*ann

现给出矩阵的大小 n（n 行 n 列），矩阵 matrix
tips：题目保证输入规范且矩阵不会太大
（尽量将两种都写出来）

示例:

输入:
3
1 2 3
4 5 6
7 8 9

输出:
0

10、迷宫问题

现给定一个 n*n 的迷宫，用 0 表示可以通行，用 1 表示障碍，要求找到从起点[0,0]到终点[n-1,n-1]的所有路径，若无路径则输出”no path”。

输入：
n　　　//迷宫大小
matrix　　//迷宫
输出：
[x0,y0] -> [x1,y1] -> END　或　no path

示例 1：

```
5
0 0 0 1 1
1 1 0 0 0
1 0 0 1 0
1 0 1 1 0
1 0 0 0 0
[0,0] -> [0,1] -> [0,2] -> [1,2] -> [2,2] -> [2,1] -> [3,1] -> [4,1] -> [4,2] -> [4,3] -> [5,5] -> END
[0,0] -> [0,1] -> [0,2] -> [1,2] -> [1,3] -> [1,4] -> [2,4] -> [3,4] -> [5,5] -> END
```

示例 2：

```
6
0 0 0 0 1 1
1 1 1 0 0 1
1 0 1 0 1 1
1 0 0 0 0 0
1 0 1 1 1 1
1 0 0 1 1 0
no path
```