

# 题目 A

## 汽水瓶

有这样一道智力题：“某商店规定：三个空汽水瓶可以换一瓶汽水。小张手上有十个空汽水瓶，她最多可以换多少瓶汽水喝？”答案是 5 瓶，方法如下：先用 9 个空瓶子换 3 瓶汽水，喝掉 3 瓶满的，喝完以后 4 个空瓶子，用 3 个再换一瓶，喝掉这瓶满的，这时候剩 2 个空瓶子。然后你让老板先借给你一瓶汽水，喝掉这瓶满的，喝完以后用 3 个空瓶子换一瓶满的还给老板。如果小张手上有  $n$  个空汽水瓶，最多可以换多少瓶汽水喝？

### 输入

输入文件最多包含 10 组测试数据，每个数据占一行，仅包含一个正整数  $n$  ( $1 \leq n \leq 100$ )，表示小张手上的空汽水瓶数。 $n=0$  表示输入结束，你的程序不应当处理这一行。

### 输出

对于每组测试数据，输出一行，表示最多可以喝的汽水瓶数。如果一瓶也喝不到，输出 0。

### 样例输入

### 样例输出

3	1
10	5
81	40
0	

# 题目 B

## 弟弟的作业

你的弟弟刚做完了“100 以内数的加减法”这部分的作业，请你帮他检查一下。每道题目（包括弟弟的答案）的格式为  $a+b=c$  或者  $a-b=c$ ，其中  $a$  和  $b$  是作业中给出的，均为不超过 100 的非负整数； $c$  是弟弟算出的答案，可能是不超过 200 的非负整数，也可能是单个字符“?”，表示他不会算。

### 输入

输入文件包含不超过 100 行，以文件结束符结尾。每行包含一道题目，格式保证符合上述规定，且不包含任何空白字符。输入的所有整数均不含前导 0。

### 输出

输出仅一行，包含一个非负整数，即弟弟答对的题目数量。

### 样例输入

### 样例输出

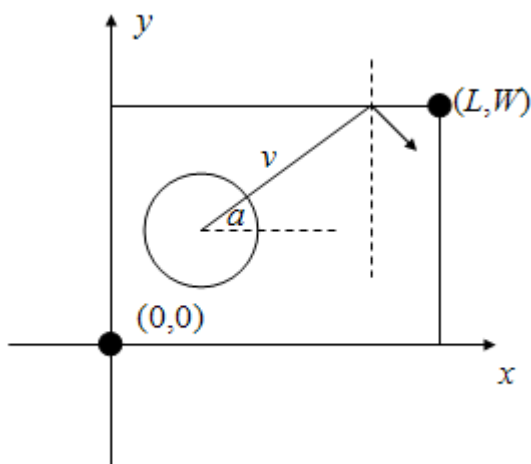
1+2=3 3-1=5 6+7=? 99-0=99	2
------------------------------------	---

[illegible]

## 题目 D

### 台球碰撞

在平面直角坐标系下，台球桌是一个左下角在 $(0,0)$ ，右上角在 $(L,W)$ 的矩形。有一个球心在 $(x,y)$ ，半径为 $R$ 的圆形母球放在台球桌上（整个球都在台球桌内）。受撞击后，球沿极角为 $a$ 的射线（即： $x$ 正半轴逆时针旋转到此射线的角度为 $a$ ）飞出，每次碰到球桌时均发生完全弹性碰撞（球的速率不变，反射角等于入射角）。



如果球的速率为 $v$ ， $s$ 个时间单位之后球心在什么地方？

#### 输入

输入文件最多包含 25 组测试数据，每个数据仅一行，包含 8 个正整数  $L, W, x, y, R, a, v, s$  ( $100 \leq L, W \leq 10^5$ ,  $1 \leq R \leq 5$ ,  $R \leq x \leq L-R$ ,  $R \leq y \leq W-R$ ,  $0 \leq a < 360$ ,  $1 \leq v, s \leq 10^5$ )，含义见题目描述。 $L=W=x=y=R=a=v=s=0$  表示输入结束，你的程序不应当处理这一行。

#### 输出

对于每组数据，输出仅一行，包含两个实数  $x, y$ ，表明球心坐标为 $(x,y)$ 。 $x$ 和 $y$ 应四舍五入保留两位小数。

#### 样例输入

```
100 100 80 10 5 90 2 23
110 100 70 10 5 180 1 9999
0 0 0 0 0 0 0 0
```

#### 样例输出

```
80.00 56.00
71.00 10.00
```

# 题目 E

## 内部收益率

在金融中，我们有时会用内部收益率  $IRR$  来评价项目的投资财务效益，它等于使得投资净现值  $NPV$  等于 0 的贴现率。换句话说，给定项目的期数  $T$ 、初始现金流  $CF_0$  和项目各期的现金流  $CF_1, CF_2, \dots, CF_T$ ， $IRR$  是下面方程的解：

$$NPV = CF_0 + \frac{CF_1}{1 + IRR} + \frac{CF_2}{(1 + IRR)^2} + \dots + \frac{CF_T}{(1 + IRR)^T} = 0$$

为了简单起见，本题假定：除了项目启动时有一笔投入（即初始现金流  $CF_0 < 0$ ）之外，其余各期均能赚钱（即对于所有  $i=1,2,\dots,T$ ， $CF_i > 0$ ）。根据定义， $IRR$  可以是负数，但不能大于 -1。

### 输入

输入文件最多包含 25 组测试数据，每个数据占两行，第一行包含一个正整数  $T$  ( $1 \leq T \leq 10$ )，表示项目的期数。第二行包含  $T+1$  个整数： $CF_0, CF_1, CF_2, \dots, CF_T$ ，其中  $CF_0 < 0, 0 < CF_i < 10000$  ( $i=1,2,\dots,T$ )。  $T=0$  表示输入结束，你的程序不应当处理这一行。

### 输出

对于每组数据，输出仅一行，即项目的  $IRR$ ，四舍五入保留小数点后两位。如果  $IRR$  不存在，输出 "No"，如果有多个不同  $IRR$  满足条件，输出 "Too many"（均不含引号）

### 样例输入

```
1
-1 2
2
-8 6 9
0
```

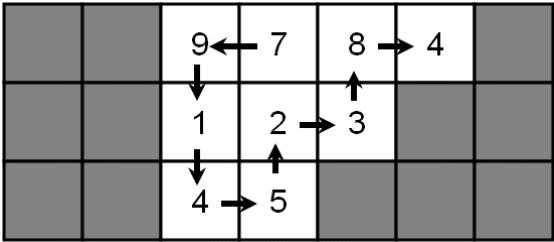
### 样例输出

```
1.00
0.50
```

# Problem F

## Biggest Number

You have a maze with obstacles and non-zero digits in it:



You can start from any square, walk in the maze, and finally stop at some square. Each step, you may only walk into one of the ***four neighbouring squares (up, down, left, right)*** and you cannot walk into obstacles or walk into a square more than once. When you finish, you can get a number by writing down the digits you encounter in the same order as you meet them. For example, you can get numbers 9784, 4832145 etc. The biggest number you can get is 791452384, shown in the picture above.

Your task is to find the biggest number you can get.

### Input

There will be at most 25 test cases. Each test begins with two integers  $R$  and  $C$  ( $2 \leq R, C \leq 15, R * C \leq 30$ ), the number of rows and columns of the maze. The next  $R$  rows represent the maze. Each line contains exactly  $C$  characters (without leading or trailing spaces), each of them will be either '#' or one of the nine non-zero digits. There will be at least one non-obstacle squares (i.e. squares with a non-zero digit in it) in the maze. The input is terminated by a test case with  $R=C=0$ , you should not process it.

### Output

For each test case, print the biggest number you can find, on a single line.

### Sample Input

```
3 7
##9784#
##123##
##45###
0 0
```

### Output for the Sample Input

```
791452384
```

# Problem G

## Repairing a Road

You live in a small town with  $R$  bidirectional roads connecting  $C$  crossings and you want to go from crossing 1 to crossing  $C$  as soon as possible. You can visit other crossings before arriving at crossing  $C$ , but it's not mandatory.

You have exactly one chance to ask your friend to repair exactly one existing road, ***from the time you leave crossing 1***. If he repairs the  $i$ -th road for  $t$  units of time, the crossing time after that would be  $v_i a_i t$ . It's not difficult to see that it takes  $v_i$  units of time to cross that road if your friend doesn't repair it.

You cannot start to cross the road when your friend is repairing it.

### Input

There will be at most 25 test cases. Each test case begins with two integers  $C$  and  $R$  ( $2 \leq C \leq 100$ ,  $1 \leq R \leq 500$ ). Each of the next  $R$  lines contains two integers  $x_i, y_i$  ( $1 \leq x_i, y_i \leq C$ ) and two positive floating-point numbers  $v_i$  and  $a_i$  ( $1 \leq v_i \leq 20, 1 \leq a_i \leq 5$ ), indicating that there is a bidirectional road connecting crossing  $x_i$  and  $y_i$ , with parameters  $v_i$  and  $a_i$  (see above). Each pair of crossings can be connected by at most one road. The input is terminated by a test case with  $C=R=0$ , you should not process it.

### Output

For each test case, print the smallest time it takes to reach crossing  $C$  from crossing 1, rounded to 3 digits after decimal point. It's always possible to reach crossing  $C$  from crossing 1.

#### Sample Input

```
3 2
1 2 1.5 1.8
2 3 2.0 1.5
2 1
1 2 2.0 1.8
0 0
```

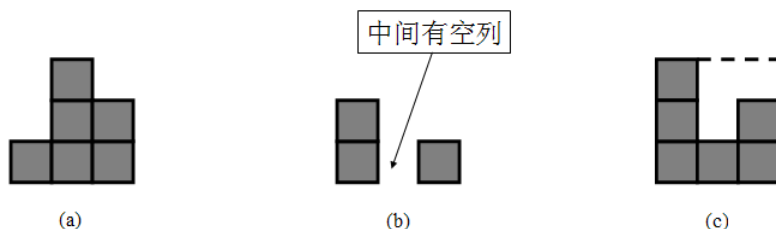
#### Output for the Sample Input

```
2.589
1.976
```

# 题目 H


## 射击游戏

A 和 B 在玩一个射击游戏，战场由若干单位正方形积木组成。积木占据了连续的若干列，且图形周长等于它最小包围矩形的周长。下图(a)是一个合法的战场，但(b)和(c)都不是：(b)中有空列；(c)的图形周长为 14，而最小包围矩形（用虚线画出）的周长为 12。受重力影响，每个积木的正下方要么是地面，要么是另一个积木。为了让战场看上去错落有致、玩着更刺激，它不能恰好是一个矩形（即：不能每列积木都一样高）。



游戏规则如下：

- 1、A 和 B 轮流射击，A 先射击。
- 2、每次射击时，首先选择一行（该行必须至少有一个积木），以及“左”和“右”中的一个方向，然后往这个方向开火。子弹的威力为 1~3 的均匀随机整数（即：威力为 1、2、3 的概率各为 1/3），表示子弹能打掉的积木个数，被打掉的积木将直接从战场中消失。如果该行的积木个数小于威力值，则子弹将在打掉该行所有积木后消失。例如，若选择往右射击从下往上数第 3 行，且威力为 2，且这一行一共有 4 个积木，则最左边的两个积木将被打掉。注意：这两个积木可以不连续。
- 3、每次射击完成后，悬空的积木垂直往下落。所有积木不再下落后，下一位选手才能开始射击。
- 4、谁打掉了最后一个积木，谁就获胜。

假定开局是 ，根据规则 1，A 先开火。射击后，B 可能面临的后续局面中的其中三个如下表：

行编号（从下往上数）	子弹前进方向	威力（随机值）	刚射击后	积木稳定后
2	从右往左	1		（同左图）
1	从右往左	2		
1	从左往右	3		

假定 A 和 B 都足够聪明，采取让自己获胜概率尽量高的策略，你的任务是计算出 A 获胜的概率。

### 输入

输入文件最多包含 25 组测试数据，每个数据仅包含两行，第一行是整数  $n$  ( $1 \leq n \leq 6$ )，即积木的列数。第二行包含  $n$  个正整数  $h_1, h_2, \dots, h_n$  ( $1 \leq h_i \leq 6$ )，表示从左往右数第  $i$  列的高度。积木的排列方式保证符合题目描述（即：图形周长等于它最小包围矩形的周长，且各列的高度不全相同）。 $n=0$  表示输入结束，你的程序不应当处理这一行。

### 输出

对于每组数据，输出仅一行，即 A 获胜的概率，四舍五入保留六位小数。

### 样例输入

### 样例输出

3	0.555556
2 1 1	
0	





# 题目 I

## 战场的数目

在上题中，假设战场的图形周长为  $p$ ，一共有多少种可能的战场？

例如， $p < 8$  时没有符合要求的战场， $p = 8$  时有 2 种战场：



$p = 10$  有 9 种战场：



要求输出方案总数模 987654321 的值。

### 输入

输入文件最多包含 25 组测试数据，每个数据仅包含一行，有一个整数  $p$  ( $1 \leq p \leq 10^9$ )，表示战场的图形周长。 $p = 0$  表示输入结束，你的程序不应当处理这一行。

### 输出

对于每组数据，输出仅一行，即满足条件的战场总数除以 987654321 的余数。

### 样例输入

### 样例输出

7	0
8	2
9	0
10	9
0	

# Problem J

## Infinite Dictionaries

A dictionary is a set of key-value pairs, for example:

```
{'color':'red', 'price':2, 7:'test', 100:-100}
```

As you can see, **keys and values can be strings or integers**. What's more, values can also be dictionaries or variable references. Here is the formal definition of terms that will be used soon:

```
key    ::=      INTEGER | STRING
value  ::=      INTEGER | STRING | dict
pair   ::=      key ':' value
dict   ::=      '{' [pair (',' pair)*] '}'
var    ::=      'a'|'b'|'c'|...|'z'
slot   ::=      var([' key '])*
lvar   ::=      slot
rvar   ::=      slot | value
```

Here ('[' key '])\* means zero or more subscripts, [pair (',' pair)\*] means zero or more key-value pairs.

Strings are always enclosed by single quotes (') and consists of up to 10 lower-case letters. Integers always have absolute values of no more than 1000. You can insert spaces anywhere, except inside strings or integers. For example, { 'a':-1} and { 'a' : -1 } are the same, but { 'a b':1} and { 'a':- 1} are both illegal.

Your task is to execute a series of commands and print the results. There are 3 kinds of commands:

**1. Assignment.** <lvar> = <rval>

After assigning a slot to a slot (rather than a value), the left-hand slot will be holding a reference to the right-hand. For example, After executing the following commands, b[1][0] is 1, rather than 0:

```
a = {0:0}
b = {}
b[1] = a
a[0] = 1
```

Slots must be assigned before it is *read* or *subscripted*, and integers and strings cannot be subscripted. Consider the following comammd list:

```
c = {}
c[0] = 3
c[1] = c[0]
d[0] = 'i'
c = d
d = c[1]['a']
c[2][2] = 2
```

The first three commands are legal, but the next two are both illegal because slot d must be assigned before it is read or subscripted. The last three are also illegal.

**2. Length:** length(<slot>)

Output the number of key-value pairs in the slot. Note that nested pairs are not counted. For example:

```
a = {0: {0:0, 1:1}}
length(a)
```

will output 1, not 3. In this command, it is guaranteed that <slot> is storing a dictionary, not a string or an integer.

**3. Infinity test:** `test(<slot>)`

If the slot can be subscripted indefinitely, output 1. Otherwise, output 0. For example, after executing the following command list:

```
d = {}
d[0] = d
```

Then `d` is infinite, since `d[0][0][0][0][0][0]...` is always `d`. In this command, it is guaranteed that `<slot>` is storing a dictionary, not a string or an integer.

**Input**

The input contains at most 10000 lines of commands, each line will be non-empty and will contain no more than 300 characters. All the commands are legal.

**Output**

Print the output (one line for each length/test command).

**Sample Input**

```
c = {}
d = {'color': 'red', 'price': 2, 7:
'test', 100: -100}
length(d)
d[7] = {'this': 'is', 'a': 'book'}
length(d)
d[8] = {'this' : 'is', 'another' :
{'a' : 'book', 'b': 'book2'} }
length(d)
c[7] = c
test(c)
test(d)
length(c)
d[0] = c
length(d)
test(d[0])
```

**Output for the Sample Input**

```
4
4
5
1
0
1
6
1
```

**Notes**

The term definitions in this problem use a modified BNF grammar notation that is described in the official documentation of the Python programming language. Here is the explanation of the notation, extracted from the documentation:

*Each rule begins with a name (which is the name defined by the rule) and ::= . A vertical bar (|) is used to separate alternatives; it is the least binding operator in this notation. A star (\*) means zero or more repetitions of the preceding item; likewise, a plus (+) means one or more repetitions, and a phrase enclosed in square brackets ([]) means zero or one occurrences (in other words, the enclosed phrase is optional). The \* and + operators bind as tightly as possible; parentheses are used for grouping. Literal strings are enclosed in quotes. White space is only meaningful to separate tokens. Rules are normally contained on a single line; rules with many alternatives may be formatted alternatively with each line after the first beginning with a vertical bar.*

# Problem K

## Tetrahedrons and Spheres

There are  $a$  tetrahedrons and  $b$  spheres in the 3D-space, you're asked to calculate the volume occupied by at least one of them (i.e. volume of the union of the objects).

### Input

There will be at most 20 test cases. Each case begins with two integers  $a, b$ , the number of tetrahedrons and the number of spheres ( $1 \leq a, b \leq 5$ ). The next  $a$  lines each contains 12 integers:  $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, x_4, y_4, z_4$ , the coordinates  $(x_i, y_i, z_i)$  ( $1 \leq i \leq 4$ ) of the four vertices of a tetrahedron. The next  $b$  lines each contains 4 integers  $x, y, z, r$ , the coordinates of the center  $(x, y, z)$  and the radius  $r$  ( $r \leq 3$ ). All the coordinate values are integers with absolute values no more than 5. The input is terminated by  $a=b=0$ .

### Output

For each test case, print a single line, the volume occupied by at least one of them, rounded to three decimal points.

#### Sample Input

```
1 1
0 0 4 1 0 4 0 1 4 0
0 5
0 0 0 1
0 0
```

#### Output for the Sample Input

```
4.356
```