Problem A. Airplane

Input file:	`standard input`
Output file:	`standard output`

n people board an airplane with n seats. The first passenger has lost his boarding pass, so he sits in a random seat. Each subsequent passenger sits in his own seat if it's available or takes a random unoccupied seat if it's not.

What's the probability that the nth passenger finds his seat occupied?

Input

The input file contains several test cases. Each test case is described with one integer n on a single line ($2 \leq n \leq 1\,000$).

The last line contains a single `0` and should not be processed.

Output

For each test case, output the probability that the nth passenger finds his seat occupied on a single line.

If the probability is 0, output `0/1`. Otherwise, the probability should be expressed as an irreducible fraction a/b, where a and b are positive integers and a and b be are relatively prime. Do not print any spaces between the numbers or the division sign.

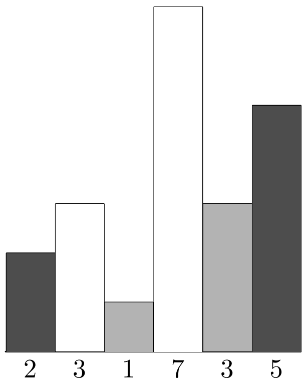
Sample input and output

standard input	standard output
2 0	1/2

Problem B. Rectangle

Input file:	`standard input`
Output file:	`standard output`

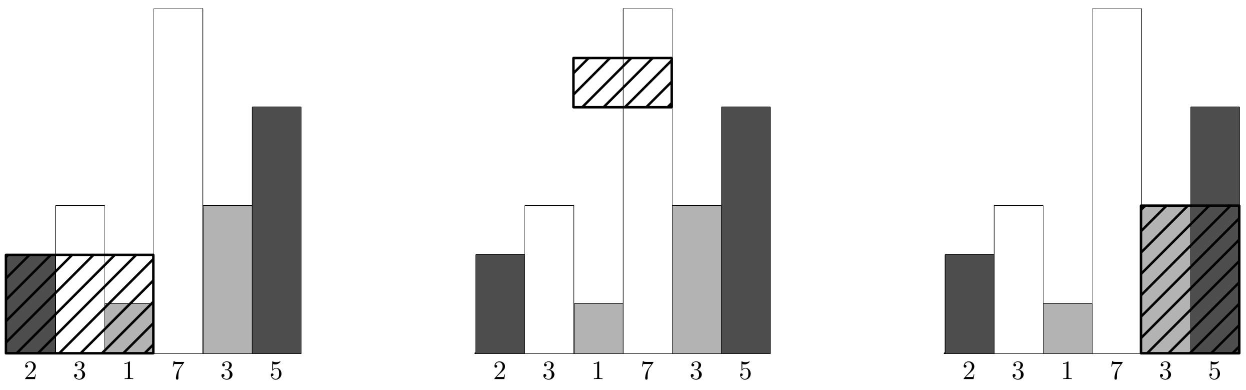
A *histogram* is a polygon composed of a sequence of rectangular bars aligned at a common base line. All bars have equal widths but may have different integer heights. Additionally, each bar is fully colored with a single color. For example, the figure below shows the histogram that consists of bars with the heights 2, 3, 1, 7, 1, 3, 5, measured in units where 1 is the width of each bar. In this case, every bar is colored with one out of three colors:



There are a lot of rectangles that can be placed inside this histogram (infinitely many, in fact). We will say that a rectangle is *nice* if and only if:

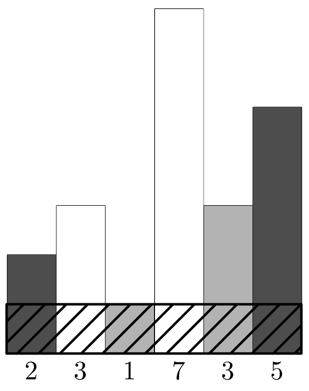
- It lies completely inside the histogram.
- It covers one or several regions of positive area for every color in the histogram (barely touching an area is not enough; the total area covered for each color must be strictly positive).

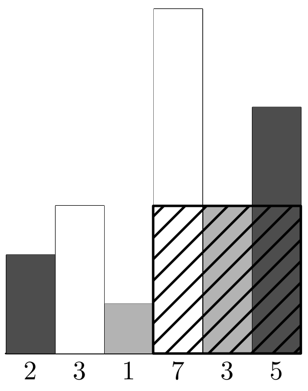
Here are some examples of rectangles that are not nice:

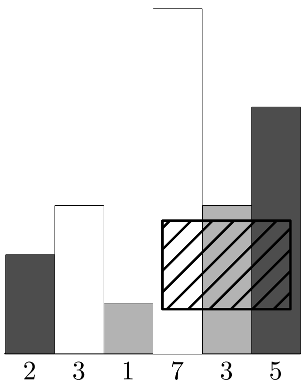


The rectangles in the two leftmost figures are not nice because they don't lie completely inside the histogram. The rectangle in the rightmost figure is not nice because the total white area covered is not strictly positive, and a nice rectangle must cover a positive area of every color.

On the other hand, here are some examples of nice rectangles:







The area of the nice rectangle in the leftmost figure is $6 \times 1 = 6$ square units, the area of the nice rectangle in the middle figure is $3 \times 3 = 9$ square units and the area of the nice rectangle in the rightmost figure is $1.8 \times 2.6 = 4.68$ square units.

Clearly, the area of the rectangle in the middle figure is largest. In fact, it turns out it doesn't exist a nice rectangle for this histogram with a larger area!

In this problem, you are given a histogram and your task is to compute the maximum area of all the possible nice rectangles. By the way, it is not hard to prove that this area will always be an integer.

Input

The input contains several test cases (at most 50).

Each test case is described by several lines. The first line contains two integer $\mathcal{N}$ and $\mathcal{C}$, the number of bars and the total number of colors in the histogram, respectively ($1 \leq \mathcal{N} \leq 10^5$ and $1 \leq \mathcal{C} \leq min(30, \mathcal{N})$).

The following line contains $\mathcal{N}$ integers h_i, the height of the ith bar from left to right ($1 \leq h_i \leq 10^9$). The following line contains $\mathcal{N}$ integers c_i, the color of the ith bar from left to right. Each color is represented as an integer between 0 and $\mathcal{C} - 1$, inclusive. You can assume that for each histogram there will be at least one bar of every color.

The last line of the input contains two zeros and should not be processed.

Output

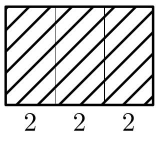
For each test case, output a single integer on a single line — the area of the largest nice rectangle for this histogram.

Sample input and output

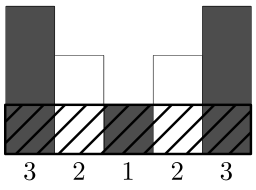
standard input	standard output
6 3	9
2 3 1 7 3 5	6
2 0 1 0 1 2	5
3 1	12
2 2 2	10
0 0 0	10
5 2	2999999991
3 2 1 2 3	
1 0 1 0 1	
6 4	
1 2 3 4 5 6	
0 1 2 3 1 0	
7 2	
1 2 3 4 3 2 1	
0 1 0 1 0 1 0	
10 2	
2 1 2 1 1 2 1 2 2 1	
1 0 0 0 1 0 0 1 1 0	
3 2	
1000000000 999999997 999999999	
0 1 1	
0 0	

Explanation of the sample cases

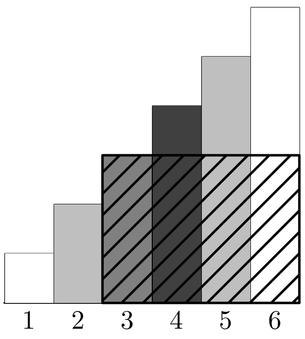
Below are some of the sample test cases with their respective largest nice rectangles:



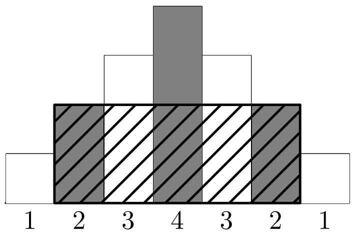
Example 2



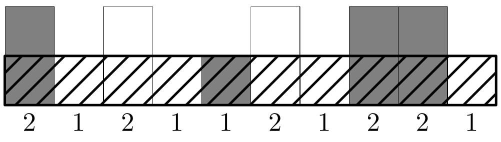
Example 3



Example 4



Example 5



Example 6

Problem C. Little Nephew

Input file:	`standard input`
Output file:	`standard output`

Your little nephew is 5 years old. He tells you he has a hats of different colors, b T-shirts of different colors, c pants of different colors, d pairs of socks of different colors and e pairs of shoes of different colors (you know how kids are). He is on that age where kids ask and ask questions all day long. He asked you in how many different ways he could possibly get dressed.

Since he is just 5, sometimes he wears socks or shoes of different colors. For example, he might wear a red sock on his right foot and a blue sock on his left foot. We will consider two ways of dressing different if they differ in the color of the hat, the color of the T-shirt, the color of the pants, the color of the sock on the left foot, the color of the sock on the right foot, the color of the shoe on the left foot or the color of the shoe on the right foot.

He's clever enough to figure out which is the left shoe and which is the right shoe, so he'll never put the wrong shoe on the wrong foot like you used to do. Also, he **always** wears a hat, a T-shirt, a pair of pants, a pair of socks and a pair of shoes (I guess someone told him it's not good to go walking around there half-naked).

Here's an example. If he has one red hat, two T-shirts (one blue and one green), one pair of pants (black), two pairs of socks (one white pair and one orange pair) and two pairs of shoes (one brown pair and the other one yellow) then he can dress in 32 different ways. Here are some of the valid different ways how he can dress (the changes between consecutive ways in this list are shown in **bold**):

- The red hat, the blue T-shirt, the black pants, a white sock on the left foot, a white sock on the right foot, a brown shoe on the left foot, a brown shoe on the right foot.
- The red hat, the **green** T-shirt, the black pants, a white sock on the left foot, an **orange** sock on the right foot, a brown shoe on the left foot, a brown shoe on the right foot.
- The red hat, the green T-shirt, the black pants, an **orange** sock on the left foot, a **white** sock on the right foot, a brown shoe on the left foot, a brown shoe on the right foot.
- The red hat, the green T-shirt, the black pants, an orange sock on the left foot, a white sock on the right foot, a brown shoe on the left foot, a **white** shoe on the right foot.

Notice that the only difference between the second and the third way is that he swapped the color of his left and right socks. These are two different ways of getting dressed according to the rules above.

He said that if you wrote a program that could calculate this number he would stop asking questions!

Input

The input contains several test cases.

Each test case is described by five integers a, b, c, d and e ($1 \leq a, b, c, d, e \leq 20$). These integers represent the values explained above and will be separated by one or more spaces.

The last line of the input contains five 0's and should not be processed.

Output

For each test case, output the number of different ways in which the kid can get dressed. Output this number on a single line.

Sample input and output

standard input	standard output

1 2 1 2 2
1 1 1 1 1
12 3 18 2 19
3 4 16 1 4
0 0 0 0 0
	32
	1
	935712
	3072

Problem D. Professor Lazy, Ph.D.

Input file:	`standard input`
Output file:	`standard output`

Professors are very motivated when they have to travel abroad for a conference (of course, if fees are paid by the university), but they don't have the same attitude when the moment to grade exams arrives.

Professor Lazy, Ph.D., has a particular way to grade exams (and very unfair, by the way). He puts all his exams in a box and then starts getting them out one by one in a totally random fashion. He assigns grade α to the first exam that he gets out of the box, and grade β to the second exam that he gets out of the box. From that point on, he assigns a grade to each of the exams based on the grades of the previous two exams. What he does is that he takes the grade of the immediately previous exam, adds 1 and divides by the grade of the exam before the previous one.

For example, let's imagine that $\alpha = 2$ and $\beta = 3$. This is what happens:

- The first exam gets $\alpha = 2$.
- The second exam gets $\beta = 3$.
- The previous two grades are α and β, so the third exam gets $\frac{(1+\beta)}{\alpha} = \frac{(1+3)}{2} = 2$.
- The previous two grades are β and $\frac{(1+\beta)}{\alpha}$, so the fourth exam gets $\frac{1+\frac{(1+\beta)}{\alpha}}{\beta} = \frac{1+2}{3} = 1$.
- The procedure continues until he's done with all exams.

More formally, we can define the grade Q_n of the nth exam with a recurrence like this:

$$Q_n = \begin{cases} \alpha & \text{if } n = 0, \\ \beta & \text{if } n = 1, \\ \dfrac{1+Q_{n-1}}{Q_{n-2}} & \text{if } n \geq 2. \end{cases}$$

Even this simple procedure is a lot of work for Professor Lazy, Ph.D., so he asks you to write a program to do it for him. He wants to spend all day long drinking coffee in the cafeteria with other professors. Given α, β and n find the value of Q_n.

Note that the grades do not necessarily lie inside a fixed range. They are just arbitrary integers.

Input

The input contains several test cases (at most 1000). Each test case is described by three integer numbers α, β and n on a single line ($1 \leq \alpha, \beta \leq 10^9$ and $0 \leq n \leq 10^{15}$).

The last line of the input contains three zeros and should not be processed.

Output

For each test case, write the value of Q_n in a single line. The input will be such that the value of Q_n is always an integer. Furthermore, Q_i will never be zero for $0 \leq i \leq n$ (in other words, division by zero will never arise when evaluating the recurrence).

Sample input and output

standard input	standard output
1 1 0	1
1 2 1	2
5 9 2	2
2 3 3	1
7 4 4	2
2109 650790 344341899059516	650790
45861686 57 594261603792314	804591
2309734 21045930 808597262407955	2309734
0 0 0	

Problem E. The Turanga Leela Problem

Input file:	`standard input`
Output file:	`standard output`

It's been a few years since Bender solved his famous bending problem. Today, Turanga Leela, son of Morris and Munda and captain of the Planet Express spaceship, is playing a game with Bender and Philip J. Fry.

Everybody knows that Leela is cyclop. When Leela was still an infant, her parents gave her up to an orphanarium with a note scribbled with mysterious symbols to make an impression that Leela was an alien, so that she would have a better life than a typical mutant. But other kids used to make fun of her only eye, so Leela had a very perturbing childhood.

Turanga Leela, from Futurama

Bender and Philip J. Fry are making fun of Leela, because apparently she doesn't know how to count (they say her parents didn't know how to count either, and that's why she has only one eye. Then they burst out in explosive laughter). She insists that she knows how to count, but secretly she needs your help.

Philip J. Fry chooses a positive integer a, and Bender chooses a positive integer b. Leela is supposed to count the number of positive integers m such that a and b leave the same remainder when divided by m. In other words, she's supposed to find the size of the set $\{m \in \mathbb{Z}_{>0} \mid a \equiv b \pmod{m}\}$.

But she's totally clueless on how to proceed. Help her, please!

Input

The input file contains several test cases (at most 150). Each test case is described with two different integers a and b on a single line ($1 \le a, b \le 10^9$).

The last line contains two zeros and should not be processed.

Output

For each test case, output one integer on a single line — the number of positive integers m such that a and b leave the same remainer when divided by m.

Sample input and output

standard input	standard output

2 3
2 4
5 7
4 8
100 205
33043387 255936619
0 0
	1
	2
	2
	3
	8
	40

Explanation of the sample cases

- 2 and 3 only leave the same remainder when divided by 1.
- 2 and 4 leave the same remainder when divided by 1 or 2.
- 5 and 7 leave the same remainder when divided by 1 or 2.
- 4 and 8 leave the same remainder when divided by 1, 2 or 4.
- 100 and 205 leave the same remainder when divided by 1, 3, 5, 7, 15, 21, 35 or 105.
- 33043387 and 255936619 leave the same remainder when divided by 1, 2, 3, 4, 6, 8, 12, 16, 24, 48, 569, 1138, 1707, 2276, 3414, 4552, 6828, 8161, 9104, 13656, 16322, 24483, 27312, 32644, 48966, 65288, 97932, 130576, 195864, 391728, 4643609, 9287218, 13930827, 18574436, 27861654, 37148872, 55723308, 74297744, 111446616 or 222893232.

Problem F. Ancestors

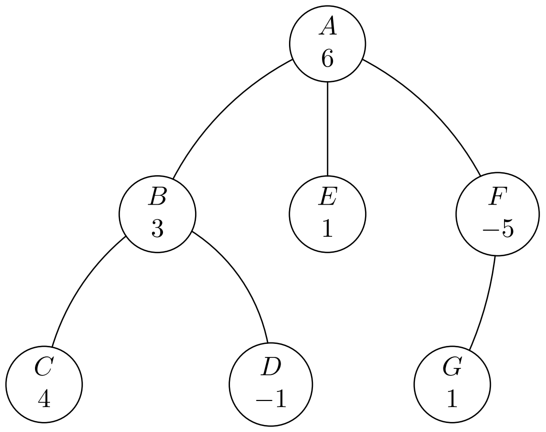
Input file:	`standard input`
Output file:	`standard output`

You will be given a connected undirected tree. Each node will have an associated integer that we will call its *value*. We also define the *value of a subset of nodes* as the sum of values of the nodes in the subset.

Consider the subsets of nodes of this tree whose size is between 1 and $\mathcal{K}$, inclusive, and satisfy the property that within each subset no node is an ancestor[1] of another. We will call these subsets the *$\mathcal{K}$-anti-ancestral subsets* of the tree.

Your task is, given the tree, to find the $\mathcal{K}$-anti-ancestral subset of maximum value.

As an example, consider the tree below. Each node is named with a capital letter and the value of each node is the integer below the letter:



Suppose $\mathcal{K} = 3$. Then the subset of nodes $\{A\}$, $\{B, E\}$, $\{C, E\}$, $\{D, G\}$ and $\{B, E, F\}$ are some of the $\mathcal{K}$-anti-ancestral subsets of the tree, because they all have between 1 and $\mathcal{K} = 3$ nodes and within each subset there doesn't exist a pair of nodes such that one is an ancestor of the other.

On the other hand, $\{A, B\}$ is not a $\mathcal{K}$-anti-ancestral subset because A is an ancestor of B, $\{A, G\}$ isn't one because A is an ancestor of G and $\{C, B, D\}$ isn't one either because B is an ancestor of both C and D. The subset $\{C, D, E, F\}$, even though it doesn't contain a node that is an ancestor of another one, is not a $\mathcal{K}$-anti-ancestral subset because it has $4 > \mathcal{K} = 3$ nodes. Similarly, the empty set $\emptyset$ is not one either because it contains 0 elements.

The value of the $\mathcal{K}$-anti-ancestral subset $\{A\}$ is 6; the value of $\{B, E\}$ is $3 + 1 = 4$; the value of $\{C, E\}$ is $4 + 1 = 5$; the value of $\{D, G\}$ is $-1 + 1 = 0$ and the value of $\{B, E, F\}$ is $3 + 1 + (-5) = -2$. Since the tree is small, it's easy to convince yourself by inspection that the maximum possible value of any $\mathcal{K}$-anti-ancestral subset is 6. However, notice that there might be several ways to achieve the maximum value: $\{A\}$ and $\{C, E, G\}$ both have value 6. You are only asked to find the value of the maximum $\mathcal{K}$-anti-ancestral subset so it doesn't really matter how many of them exist.

Input

The input file contains several test cases (at most 50).

Each test case is described on three lines:

- The first line contains two integers N and $\mathcal{K}$, the number of nodes in the tree and the parameter $\mathcal{K}$ as explained above. Assume $2 \le N \le 10^5$ and $1 \le \mathcal{K} \le 100$ (notice that these constraints guarantee that there will always exist at least one $\mathcal{K}$-anti-ancestral subset, e.g. by taking a single node). The

[1]Formally, we say that node u is an ancestor of node v if u lies on the path from v to the root of the tree.

nodes will be numbered 0 through $N-1$. The root of the tree will always be node 0. You can assume that the tree will always be connected, i.e., there will always exist a path connecting any pair of nodes.

- The second line contains N integers separated by spaces. These integers represent the values of nodes 0 through $N-1$ (the first integer is the value of node 0, the second integer is the value of node 1 and so on — the last integer is the value of node $N-1$). The value of any node will always be between -100 and 100, inclusive.
- The third line contains the description of the tree. It contains $N-1$ integers separated by spaces that represent the parent nodes of nodes 1 through $N-1$ (the first integer is the parent of node 1, the second integer is the parent of node 2 and so on — the last integer is the parent of node $N-1$). Notice that since 0 is always the root of the tree it doesn't have a parent and that's why this line contains only $N-1$ numbers instead of N.

The last line contains two zeros and should not be processed.

See the sample input for clarification about this format. The first test case is the tree given in the figure above.

Output

For each test case, output one integer on a single line — the maximum possible value of any $\mathcal{K}$-anti-ancestral subset.

Sample input and output

standard input	standard output
`7 3`	`6`
`6 3 4 -1 1 -5 1`	`1`
`0 1 1 0 0 5`	`5`
`2 1`	`8`
`-1 1`	`-27`
`0`	
`3 3`	
`1 2 3`	
`0 0`	
`8 8`	
`1 2 3 4 5 6 7 8`	
`0 1 2 3 4 5 6`	
`4 4`	
`-27 -45 -67 -98`	
`2 0 2`	
`0 0`	

Problem G. Secret word

Input file:	`standard input`
Output file:	`standard output`

Alicia and Roberto like to play games. Today, Roberto is trying to guess a secret word that Alicia chose. Alicia wrote a long string S in a piece of paper and gave Roberto the following clues:

- The secret word is a non-empty substring[2] of S (possibly equal to S).
- S starts with the secret word reversed.

Roberto knows Alicia very well, and he's sure that if there are several possible secret words that satisfy the clues above, Alicia must have chosen the longest one.

Can you help him guess the secret word?

Input

The first line of the input file contains a single integer number $T \leq 150$, the number of test cases.

T lines follow, each with a single string S. S will only contain lowercase English letters. The length of S will not exceed one million characters.

Output

For each test case, output the secret word in one line.

Sample input and output

standard input	standard output
5 colombia abcdba neversayeven neveroddoreven listentothesilence	c ba even neveroddoreven sil

Explanation of the sample cases

- `colombia`: if you take `c` and reverse it you get `c`. `colombia` starts with `c`, so this satisfies the two clues above. Furthermore, `c` is the longest word that satisfies the two clues so it must be the secret word.
- `abcdba`: if you take `ba` and reverse it you get `ab`. `abcdba` starts with `ab` and there's no longer substring that satisfies the two clues.
- `neversayeven`: if you take `even` and reverse it you get `neve`. `neversayeven` starts with `neve` and there's no longer substring that satisfies the two clues.
- `neveroddoreven`: this case is a palindrome so if we reverse it we get the same string. `neveroddoreven` starts with `neveroddoreven` (since they are the same) and obviously there's no longer substring that satisfies that, so this is the secret word.
- `listentothesilence`: Notice the secret word might be somewhere in the middle of the big word.

[2]A substring of S is defined as any **consecutive** sequence of characters belonging to S. For example, if S = `abcd` then `a`, `abcd`, `ab`, `bc` and `bcd` are some of the substrings of S but `ac`, `aa`, `aabbccdd` and `dcba` are not substrings of S.

Problem H. Zapping

Input file:	`standard input`
Output file:	`standard output`

I'm a big fan of watching TV. However, I don't like to watch a single channel; I'm constantly zapping between different channels.

My dog tried to eat my remote controller and unfortunately he partially destroyed it. The numeric buttons I used to press to quickly change channels are not working anymore. Now, I only have available two buttons to change channels: one to go up to the next channel (the $\triangle$ button) and one to go down to the previous channel (the $\bigtriangledown$ button). This is very annoying because if I'm watching channel 3 and want to change to channel 9 I have to press the $\triangle$ button 6 times!

My TV has 100 channels conveniently numbered 0 through 99. They are cyclic, in the sense that if I'm on channel 99 and press $\triangle$ I'll go to channel 0. Similarly, if I'm on channel 0 and press $\bigtriangledown$ I'll change to channel 99.

I would like a program that, given the channel I'm currently watching and the channel I would like to change to, tells me the minimum number of button presses I need to reach that channel.

Input

The input contains several test cases (at most 200).

Each test case is described by two integers a and b in a single line. a is the channel I'm currently watching and b is the channel I would like to go to ($0 \leq a, b \leq 99$).

The last line of the input contains two -1's and should not be processed.

Output

For each test case, output a single integer on a single line — the minimum number of button presses needed to reach the new channel (Remember, the only two buttons I have available are $\triangle$ and $\bigtriangledown$).

Sample input and output

standard input	standard output

`3 9`
`0 99`
`12 27`
`-1 -1`
	`6`
	`1`
	`15`

Problem I. Stones

Input file:	`standard input`
Output file:	`standard output`

Alicia and Roberto like to play games. Today, they are playing a game where there's a pile of stones on the table. The two players alternate turns removing some stones from the pile. On the first turn, a player can remove any number of stones but he or she must leave at least one stone on the table. On following turns, a player can remove at most twice the number of stones that the other player removed on the previous turn. Each player must always remove at least one stone.

The player that takes the last stone wins.

For example, let's imagine there's a pile of 8 stones on the table, and let's imagine Alicia starts.

- On the first turn, Alicia must remove a number of stones between 1 and 7 (according to the rules above, on the first turn she can remove any number of stones as long as there remains at least 1). Let's say Alicia takes 2 stones, leaving 6 on the table.
- On the second turn, Roberto must remove at least 1 stone and at most 4 stones (because 4 is twice the number of stones that Alicia removed in the previous turn). Let's say he takes 3, leaving 3 stones on the table.
- On the third turn, Alicia must remove at least 1 and at most 6 stones (6 is twice the number of stones Roberto took in the previous turn). Since only 3 stones remain on the table, she simply takes the 3 stones and wins the game!

However, Roberto could have played better. If he had taken only 1 stone on the second turn, he would have left 5 on the table with Alicia having to take between 1 and 2 stones. If Alicia took 2, Roberto could win immediately by taking the 3 stones that would remain. So Alicia's only choice is to take 1 stone, leaving 4 stones and Roberto having to take between 1 and 2 stones. Roberto would then take 1 stone, leaving 3 stones and Alicia having to take between 1 and 2 stones. Roberto could then win after Alicia's move, regardless of whether she takes 1 or 2 stones. In fact, it turns out that Roberto always has a winning strategy if there are 8 stones and Alicia plays first, even if Alicia plays in the best possible way!

Your task is to determine who wins the game if both players play optimally, assuming Alicia always plays first.

Input

The input contains several test cases.

Each test case contains a single integer n ($2 \le n \le 1000$), the number of stones that are initially on the table.

The last line of the input contains a single 0 and should not be processed.

Output

For each test case, output `Alicia` or `Roberto` on a single line, depending on who wins if both players play optimally and Alicia plays on the first turn.

Sample input and output

standard input	standard output
8 2 4 986 987 0	Roberto Roberto Alicia Alicia Roberto

Problem J. Tribonacci

Input file:	`standard input`
Output file:	`standard output`

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89. Got it?
LEONARDO PISANO BIGOLLO
13th Century

Everybody knows about Fibonacci, they guy who invented the famous sequence where the first two terms are 0 and 1 and from then on every term is the sum of the previous two.

What most people don't know is that he had a somewhat mentally retarded brother named Tribonacci. In a desperate attempt to surpass his brother and achieve eternal glory, Tribonacci invented his own sequence: the first three terms are 0, 1, 2 and from then on every term is the sum of the previous three.

Sadly, regardless of enormous courage and dedication, Tribonacci was never able to compute more than the first 3 terms of his sequence. Even more sadly, one cold night he performed an extraordinary mental effort that dilated one of the blood vessels in his brain, causing severe hemorrhage and killing him instantly. This is clinically known as an aneurysm, but of course Tribonacci did not know this (it is suspected that even pronouncing the word *aneurysm* would have been an impossible task for him).

Write a program that changes history and finds the nth term in the Tribonacci sequence modulo $1,000,000,009$.

Input

The input contains several test cases (at most 400).

Each test case contains a single integer n ($1 \le n \le 10^{16}$), the desired term in the Tribonacci sequence.

The last line of the input contains a single 0 and should not be processed.

Output

For each test case, output the nth term in the Tribonacci sequence on a single line. This number might be huge, so output the number modulo $1,000,000,009$.

Sample input and output

standard input	standard output
1 2 3 4 5 6 7 8 9 10 100 10000 10000000 100000000000 10000000000000000 0	0 1 2 3 6 11 20 37 68 125 616688122 335363379 272924536 48407255 163452242