Problem A. Fight Against Numbers

Time limit: 1 second
Memory limit: 1024 mebibytes

Star and Marco got themselves into a world where they are being attacked by swarms of numbers! Fortunately, the friends know what to do.

Each number is first served with a shot from Marco's binary blaster to expose the number's binary notation. After that, Star performs a power move one or more times: she reverses the binary notation of the number and then subtracts one. When the number becomes zero, it shatters to pieces.

The heroes have just entered the fight with a number n. How many power moves would it take to shatter this number to pieces?

Reversing works as follows: the first digit of the binary notation exchanges places with the last one, the second with the second-to-last, and so on. If leading zeroes appear in the result, they are discarded.

Input

The first line contains an integer n ($1 \le n \le 10^9$).

Output

Print a single integer: how many times would Star have to perform the power move so that the number n becomes zero and shatters to pieces.

Examples

standard input	standard output	explanation
4	1	$4_{10} = 100_2$ $100_2 \to 001_2 \to 0$
3	2	$3_{10} = 11_2$ $11_2 \to 11_2 \to 10_2$ $10_2 \to 01_2 \to 0$
11	3	$11_{10} = 1011_2$ $1011_2 \to 1101_2 \to 1100_2$ $1100_2 \to 0011_2 \to 10_2$ $10_2 \to 01_2 \to 0$

Problem B. Time of the Magical Number

Time limit: 1 second
Memory limit: 1024 mebibytes

Long ago, in the fairy kingdom of Time, there lived a kind wizard named Nikolay. He adored the magical number 239, which brought him luck and joy every time he saw it on the magical digital clock of the kingdom located at the main square.

This wonderful clock displayed time using four digits: two for the hours and two for the minutes. If the number of hours or minutes was less than 10, a 0 was placed in the front.

Nikolay can enchant time so that there will be H hours in each day, numbered from 0 to $H - 1$, and M minutes in each hour, numbered from 0 to $M - 1$. For this spell, Nikolay can select any numbers H and M between 10 and 100, inclusive.

The time shown by the clock is *lucky* only when the digits resemble the cherished combination 239:

- When the clock shows `X2:39`, where the first digit `X` can take any value from `0` to `9`.
- Or `2X:39`, where the digit `X` can take the value `0`, `2`, or `3` (because `0` does not hinder the perception of the magical number).
- Or `23:X9`, where the digit `X` can take the value `0`, `3`, or `9`.
- Or even `23:9X`, where the digit `X` can take any value from `0` to `9`.

One day, Nikolay faced the question: "Given H and M, how many minutes in a full day will the clock show a lucky number?" Help Nikolay find this number.

Input

The first line of input contains two integers H and M separated by a space: the number of hours in a day and the number of minutes in an hour ($10 \leq H, M \leq 100$).

Output

Output the number of minutes in a full day when the clock will show a lucky number.

Example

standard input	standard output
24 60	6

Problem C. Inflation

Time limit: 1 second
Memory limit: 1024 mebibytes

In a distant magical kingdom, each wizard was assigned a fixed monthly salary upon starting their service.

However, there was a problem: while the wizard diligently cast spells, prices increased over the year due to inflation.

To curb the rapid rise in prices, the wise king announced on the first day of the first month of the year the final annual inflation rate p and allowed prices to be raised only once a month: exactly one day before the wizards' salaries were paid. The wizards' salaries are paid on the last day of each month.

Moreover, for convenience, he established that there are only ten months in a year, and each month has thirty days. Each month, prices were to rise by exactly $p/10$ percent of the price of goods at the **beginning** of the year. Thus, the cost of any good increased by exactly p percent over the year.

The *current salary payment system* is structured as follows. On the first day of the first month of the year, a salary of n gold is assigned, which they will receive at the end of each month of the year. Additionally, salaries are indexed each year: if the monthly salary in a given year is n gold, and the inflation rate is p, then the monthly salary for the next year will be p percent higher, that is, it will become $n + (np/100)$ gold.

Is this an ideal solution? No! The wizards are still dissatisfied: prices rise monthly, while salaries only increase once a year. The wizards propose an alternative called the *progressive salary payment system*: on the last day of each month, the amount paid as salary has the same purchasing power that the salary of n gold had on the first day of the first month of the year. In other words, for this amount, you can buy exactly the same quantity of any goods that could be purchased for n gold on the first day of the first month.

To assess this injustice, the king has tasked you with determining, based on the wizard's monthly salary at the beginning of the year, how much money they lose over the year under the current salary payment system compared to the progressive salary payment system.

Input

The first line contains two integers n and p separated by a space: the monthly salary of a wizard in gold and the annual inflation rate ($10^3 \le n \le 10^5$; $0 \le p \le 100$).

Output

Output the amount that the wizard will lose this year compared to the progressive salary payment system. Remember that a year in the magical kingdom consists of ten months. The answer is considered correct if the absolute or relative error does not exceed 10^{-4}.

Example

standard input	standard output
4346 1	239.03

Problem D. Nine Out Of Ten

Time limit: 1 second
Memory limit: 1024 mebibytes

A mad scientist conducted n independent identical experiments and claimed that x of them were successful. It is well known that the mad scientist is wrong in exactly 90% of cases when determining the success of a single experiment. Your task is to write a program that calculates the minimum and the maximum possible number of successful experiments for all x from 0 to n. It is guaranteed that the total number of experiments is always divisible by 10.

Input

The first line contains a single integer n, which is a multiple of ten ($10 \le n \le 10\,000$).

Output

Print $n + 1$ lines. On the i-th line, output two integers separated by a space: the minimum and the maximum possible number of successful experiments for $x = i - 1$.

Example

standard input	standard output
10	9 9 8 10 7 9 6 8 5 7 4 6 3 5 2 4 1 3 0 2 1 1

Problem E. Garden Bed

Time limit: 1 second
Memory limit: 1024 mebibytes

In a wonderful land where the sun shines brighter and flowers grow more lushly, there lived a kind gardener named Nikolay. He eagerly awaited the time to open the gardening season and engage in his favorite activity: cultivating garden beds.

One day, gathering his strength, Nikolay brought a few magical pairs of boards with him. Boards from the same pair had the same length, whereas boards from different pairs may have had different lengths. Nikolay dreamed of creating a *beautiful* rectangular garden bed where the prettiest vegetables and flowers would grow. For that, he needed to use all his boards without cutting them, placing paired boards on the opposite sides of the rectangle.

And so, standing in front of his garden, Nikolay pondered: "What can be the area of a beautiful garden bed?" Help him determine the minimum and maximum positive area he could obtain.

Input

The first line contains an integer n: the number of pairs of boards ($2 \le n \le 7$).

The second line contains n integers: the lengths of boards in the first, second, ..., n-th pair. Each length is an integer from 1 to 10^8.

Output

Output two integers: the minimum and maximum area of a beautiful garden bed.

Examples

standard input	standard output
2 10 239	2390 2390
3 1 2 2	4 6

Problem F. Largest Area

Time limit: 1 second
Memory limit: 1024 mebibytes

You are given a rectangle on the plane with vertices at integer points. Calculate the area of the largest ellipse that can fit entirely within this rectangle. Recall that a *rectangle* is a convex quadrilateral with all right angles, and an *ellipse* is a figure on the plane defined by two focal points F_1 and F_2 (not necessarily with integer coordinates; possibly $F_1 = F_2$) and a real number $d > |F_1F_2|$, consisting of points P on the plane such that $|F_1P| + |F_2P| \le d$.

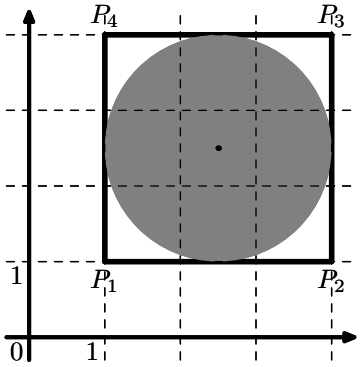
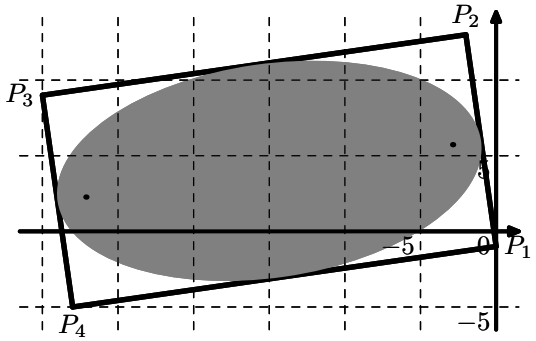
Input

The input consists of a single line containing eight integers $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$: the coordinates of four points ($-1000 \le x_i, y_i \le 1000$). The points (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4) are the vertices of a rectangle with a positive area, listed in counterclockwise order.

Output

Output a single real number: the largest area of an ellipse that can be drawn inside the rectangle given in the input. The answer is considered correct if its relative or absolute error does not exceed 10^{-6}.

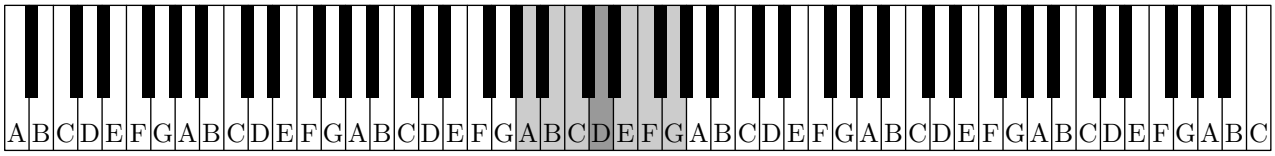
Examples

standard input	standard output	explanation
1 1 4 1 4 4 1 4	7.06858347	
0 -1 -2 13 -30 9 -28 -5	314.15926536	

Problem G. Out of Bounds Piano

Time limit: 1 second
Memory limit: 1024 mebibytes

Your friend Gassassini Pianini is learning to play the piano. He has a piano at home with a standard layout of 88 keys:



In this problem, we will only be interested in the 52 white keys. They are named consecutively with the letters of the English alphabet A–G in a cycle of length seven (called an *octave*), as shown in the image.

Gassassini Pianini has obtained the scores of several musical pieces by the violinist Paganini and wants to transpose them for the piano. For each piece, he has already decoded the musical notation and obtained a string of letters A–G that constitutes the melody. Now he wants to play the pieces on the piano, and he does not care from which octave a particular note is taken: for example, if the string indicates that the next note is A, he will be satisfied with any of the eight A notes available on the piano.

The problem is that Gassassini Pianini is still inexperienced and cannot stretch his fingers too wide. He can choose the first note freely (from the available keys with the required letter); however, for each subsequent note, he can only play one that is at most three keys away from the previous one in either direction. For example, suppose the last note played was the dark gray D in the illustration above; then the next note can either be exactly that same D or one of the six light gray notes in its vicinity.

You are given string transcriptions of several musical pieces that interest Gassassini Pianini; determine whether he will be able to play them or if he needs to practice finger positioning.

Input

The first line contains an integer t: the number of musical pieces ($1 \le t \le 10^4$).

Each of the following t lines contains a non-empty string s_i of uppercase English letters A–G: the transcription of a musical piece. The total number of letters in all transcriptions is at most $2 \cdot 10^5$.

Output

For each of the t strings, print "`Yes`" if Gassassini Pianini can play the piece on the piano, and "`No`" if it is impossible. Print each answer on a separate line. You may choose the case of the letters you print arbitrarily; in other words, the checking program does not care which of your letters are uppercase and which are lowercase.

Example

standard input	standard output
5 A ABCDEFGABCDEFGABCDE ADGCFBEADGCFBEADGCF ABEADGCFBEADGCFBEAD CCGGAAGFFEEDDCGGFFEEBGGFFEEB	Yes YES NO no YES

Problem H. Competition Results

Time limit: 1 second
Memory limit: 1024 mebibytes

There are n runners participating in a race. Each runner is assigned a unique number from 1 to n. They have arrived at the finish line in some specific order, with no ties. Let us say that runner i has performed an *upset* of runner j if i finished before j and $i < j$.

For each i from 1 to n, it is known that runner i has performed exactly a_i upsets of other runners. Your task is to restore the competition results: the number of the runner that took first place, the number of the runner that took second place, ..., the number of the runner that took the n-th place. It can be shown that the answer is always unique, assuming that it exists.

Input

The first line of the input contains an integer n from 1 to 1000: the number of runners.

The second line contains n space-delimited integers $a_1, a_2, \ldots, a_n$, where a_i is the number of upsets performed by runner i.

The given data is consistent with some possible results of the competition: for every i, it is true that $a_i \le n - i$. In particular, $a_n = 0$.

Output

Print n space-separated integers: the numbers of runners who took first, second, ..., n-th place.

Examples

standard input	standard output
5 3 0 2 1 0	3 1 4 5 2
1 0	1
2 0 0	2 1

Note

Let us check that the answer to the first example is consistent with the given numbers a_i.

1. Runner 1 has upset runners 2, 4, and 5.
2. Runner 2 took the last place and, therefore, has not outperformed anyone. Hence, runner 2 has performed no upsets.
3. The runner with the number 3 took the first place and, therefore, has upset both runners with larger numbers.
4. The runner with the number 4 has upset a single other runner: runner 5.
5. There are no runners with numbers larger than 5. Therefore, runner 5 has performed no upsets.

Problem I. Who Am I?

Time limit: 1 second
Memory limit: 1024 mebibytes

This is a run-twice problem.

A team of n players is sitting around a large round table in a library and playing a cooperative game. Each player has a card with an integer from 1 to n glued to their forehead. Players can see each other's numbers but cannot see their own numbers, which they are trying to guess. Since the game takes place in a library where everyone must be quiet, players make their guesses in writing and simultaneously, without communicating with each other. The team wins if at least one player guesses their number correctly.

Your task is to devise a winning strategy for the team. The strategy will be tested in two runs.

During the first run, you will be given the numbers on the cards of all players at the table in clockwise order. After that, you need to select the number of the player who will correctly guess their card.

During the second run, you will be informed of the number of the chosen player, as well as the numbers on the cards that this player can see from their position, from left to right. In response, you have to output the number on the chosen player's card.

Input

During both runs, the first line contains a single integer: 0 if it is the first run, or the number of the player chosen during the first run if it is the second run.

The second line contains an integer n: the number of players, or the number of cards visible to the player ($n \le 2 \cdot 10^5$). The third line contains n integers: the numbers on the cards.

There is at least one player at the table.

Output

During the first run, output a single integer: the number of the player who will correctly guess their card.

During the second run, also output a single integer: the guess of that player.

Examples

standard input	standard output
0 6 1 2 3 3 3 6	2
2 5 3 3 3 6 1	2

Problem J. Nature Reserve

Time limit: 1 second
Memory limit: 1024 mebibytes

There are unicorns living in a magical kingdom on a plane. People know the coordinates of n points on the map: the places of unicorn sightings.

Recently, the king agreed to declare the territory inhabited by unicorns a nature reserve. The boundaries of the reserve should be two parallel lines that will contain all n places of unicorn sightings between them (or on them). However, land in the kingdom is expensive, so the king does not want to allocate anything extra for the reserve: each boundary must pass through at least one of the n points.

You are the chief ecologist of the kingdom. Your task is to choose the boundaries of the reserve so that the king's will is fulfilled, but the reserve is as large as possible. The size of the reserve is defined as the distance between the boundaries.

Input

The first line contains an integer n: the number of places unicorn sightings ($2 \le n \le 1000$).

Each of the following n lines contains two integers x and y: the coordinates of a point on the map where unicorns have been seen ($0 \le x, y \le 10^4$). All given points are distinct.

Output

We will define a line by three coefficients: a, b, and c. Such a line contains points (x, y) that satisfy the equation $ax + by + c = 0$.

Output two lines specifying the borders. On each line, print three integers: a, b, and c, the coefficients of a line equation ($|a|, |b|, |c| \le 10^9$).

The distance between the lines must be the maximum possible. It can be shown that there exists an optimal answer that satisfies the above constraints. If there are several optimal answers, print any one of them.

Example

standard input	standard output
4 0 0 3 4 8 1 1 1	8 1 0 8 1 -65

Problem K. Hex Operations

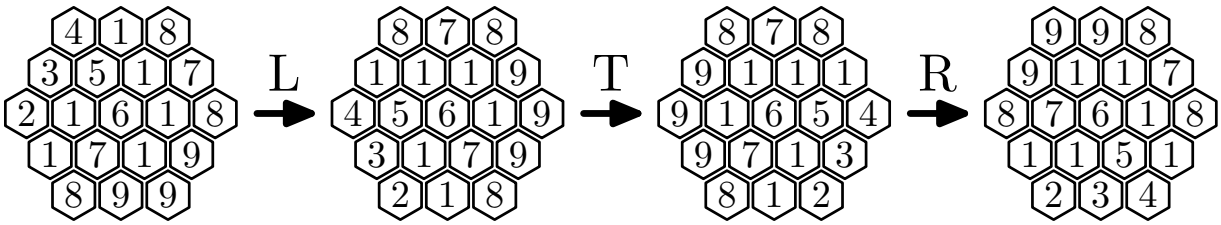
Time limit: 2 seconds
Memory limit: 1024 mebibytes

In this problem, a *hex* is a six-sided board consisting of *cells*: small regular hexagons. A hex of size n has n cells on each side. Each cell contains an integer.

Consider the following operations with a hex:

- "`T`": reflect the hex over its vertical symmetry axis,
- "`R`": rotate the hex 60 degrees clockwise,
- "`L`": rotate the hex 60 degrees counter-clockwise.

Here is an example of a hex of size $n = 3$ and operations with it:



You are given a hex and a sequence of operations. Perform all the given operations in order and print the resulting hex.

Input

The first line contains an integer n, the size of the hex ($2 \le n \le 500$). The next $2n-1$ lines contain integers from 1 to 99 initially contained in the cells of the hex. Each line contains space-separated integers: exactly as many as the number of cells in the corresponding row of the hex. Additionally, for ease of reading, these lines may contain additional spaces before all numbers, between consecutive numbers, and after all numbers.

The last line is the sequence of operations. It has a length from 1 to 250 000 characters and consists of letters "`T`", "`R`" and "`L`".

The total size of the input, including spaces and line breaks, does not exceed 2^{23} bytes.

Output

Print the hex after performing all operations: $2n - 1$ lines containing integers. Each line should contain several space-separated integers: exactly as many as the number of cells in the corresponding row of the hex. Additionally, for ease of reading, these lines may contain additional spaces before all numbers, between consecutive numbers, and after all numbers.

The total size of the output, including spaces and line breaks, should not exceed 2^{23} bytes.

Example

standard input	standard output

3
  4 1 8
 3 5 1 7
2 1 6 1 8
 1 7 1 9
  8 9 9
LTR
	9 9 8
	 9 1 1 7
	8 7 6 1 8
	 1 1 5 1
	  2 3 4

Problem L. Collatz Hypothesis and Random Increases

Time limit:	3 seconds
Memory limit:	1024 mebibytes

This is an interactive problem.

We define the Collatz function collatz(x) which operates on integers as follows: if x is even, then collatz(x) $= \frac{x}{2}$, and otherwise collatz(x) $= 3x + 1$. The famous Collatz hypothesis states that if you start with any positive integer x_0 and construct a sequence $x_1 =$ collatz(x_0), ..., $x_{i+1} =$ collatz(x_i), then the sequence will eventually reach the number one.

The incredible complexity that has kept the hypothesis neither proven nor disproven lies in the very chaotic behavior of the sequence $\{x_i\}$ until it reaches one. Even for very small numbers, one can reach the number one quite late: for example, starting with $x_0 = 9$, we get

$$9 \to 28 \to 14 \to 7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1,$$

and if we start with $x_0 = 27$, we only reach one after 111 applications of collatz(x)!

You are sitting in front of a machine that has a screen and two buttons: red and blue. The screen displays a positive integer (it is guaranteed to be a random integer from 2 to 10^7, inclusive), and you need to turn it into one. The red button is called `collatz`, and it replaces the number x on the screen with collatz(x). The blue button is called `random`, and it replaces x with a random integer from $3x + 1$ to $6x$, inclusive. Pressing the buttons is not free: after each button press, when the screen displays the number x_{i+1}, you need to insert as many tokens into the machine as there are digits in the decimal representation of the number x_{i+1}. For example, in the above process starting from nine, you need to pay for all the digits of the numbers $28, 14, 7, \ldots, 2, 1$; this will cost you 32 tokens.

If you only press the red button, you will reach one, spending an average of 707 tokens. Although the blue button always increases the number on the screen (and significantly), you can speed up the process of reaching one by pressing the blue button at the right moments! Your task is to write a program that spends on average no more than 600 tokens for a single number. To reduce the randomness in the checks, we will provide the program with $t \le 50$ random starting numbers in each test, and you need to turn all of them into ones for a total of $600 \cdot 50 = 30\,000$ tokens.

Interaction Protocol

First, read a line containing an integer t: the number of starting numbers ($1 \le t \le 50$). The jury has t starting numbers x_0, selected uniformly and independently at random from the range $[2; 10^7]$, and you need to turn them all into ones in sequence.

At the beginning of the i-th round, read a line containing a single integer x_0: the next starting number ($2 \le x_0 \le 10^7$). Then you need to transform the number. After each transformation, if the screen shows the number x_j, output a line with either the string "`collatz`" (the letters can be in any case) if you want to replace the number with $x_{j+1} =$ collatz(x_j), or the string "`random`" (the letters can be in any case) if you want to replace the number with x_{j+1} which is a random integer uniformly selected from the range $[3x_j + 1; 6x_j]$. After printing each line, do not forget to flush the output buffer, otherwise, you will likely receive an error of `Idleness Limit Exceeded`:

- `std::cout.flush()` in C++;
- `stdout.flush()` in Python;
- `System.out.flush()` in Java.

After that, read a line containing a single integer x. The following options are possible:

- if $x = 0$, you have run out of tokens;
- if $x = 1$, then $x_{j+1} = 1$: you have successfully completed the round, move on to the next round (for which you should start by reading the next starting number x_0);

- if $x > 1$, then $x_{j+1} = x$: the number on the screen has changed after pressing the button, continue outputting the strings "`collatz`" and "`random`". Note that, during the interaction, x may exceed 2^{64} as well as 2^{128}.

The jury will also send your program the number 0 if you violate the output format and print any string other than "`collatz`" and "`random`". Upon reading 0, your program should immediately terminate to receive a `Wrong Answer` verdict. Otherwise, the verdict can be anything other than `Accepted`.

After reading the t-th number one, terminate the program to receive the `Accepted` verdict.

Example

standard input	standard output
2 4 2 1 3 16 8 4 2 1	 Collatz cOLLaTZ RANDOM collatz COLlatz cOLLATz CoLlAtZ

Note

The example is the only test where the starting numbers x_0 are chosen not randomly, but manually.

Problem M. Sums of Two

Time limit: 2 seconds
Memory limit: 1024 mebibytes

Lina the Magician claims that a common modern computer can easily perform a hundred billion operations per second! To prove it, she proposes to run the following calculations.

Let V be a set of integers, initially empty. We are given the starting value of the integer s. Make n steps described below:

- $s \leftarrow (s \cdot 618\,023 + 1) \bmod 999\,983$;
- find the number of distinct pairs of integers in V that have the sum s;
- if this number is even, insert s into the set V.

How many elements will there be in V after n steps?

Formally: on each step, we count the number of pairs (a, b) where $a \in V$, $b \in V$, $a \le b$ and $a + b = s$.

Input

The first line contains integers n and s ($1 \le n \le 200\,000$; $0 \le s < 999\,983$; $s \ne 742\,681$).

Output

Print a single integer: the size of set V after n steps.

Example

standard input	standard output
4 179629	3

Note

In the example, the values of s on the four steps are 740 740, 139 655, 469 353, and 880 395.

Problem N. Wanted: Second Sock

Time limit: 1 second
Memory limit: 1024 mebibytes

Nikita is preparing for a competition. The most challenging part of this task is to find a pair of matching socks. In the drawer, there are p different pairs of socks, as well as m single socks whose pairs he has lost long ago. Nikita takes socks one by one until he finds two matching socks among those he has pulled out. What is the expected number of socks Nikita will pull out?

Calculate this expected value modulo the prime number $10^9 + 7$. Specifically, if the answer is a rational number r/q, you need to output a number a such that $0 \le a < 10^9 + 7$ and $a \cdot q = r \pmod{10^9 + 7}$.

Input

The first line contains an integer t: the number of test cases ($1 \le t \le 10^4$). The following lines describe the test cases.

Each test case is given on a separate line containing two integers p and m: the number of pairs of socks and the number of single socks in the drawer ($1 \le p \le 10^6$; $0 \le m \le 10^6$).

Output

For each test case, output a line containing a single integer: the answer to the problem.

Example

standard input	standard output
1 1 1	666666674

Note

In the first test case, the expected value is $8/3$, which equals $666\,666\,674$ modulo $10^9 + 7$.