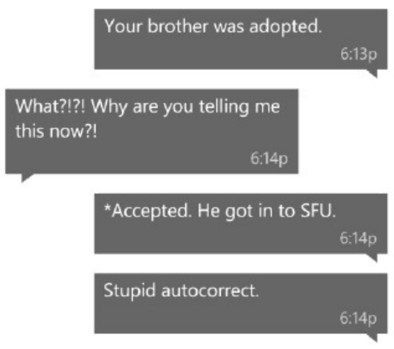
Problem A. Autocorrect

Source file name:	autocorrect.c, autocorrect.cpp, autocorrect.java, autocorrect.py
Input:	`Standard`
Output:	`Standard`

A friend's Ph.D. thesis describes a novel scheme for word prediction which helps people type faster on smartphones. You've seen obvious business potential in your friend's research so you've prototyped a virtual keyboard app using it to demonstrate the technology to venture capitalists. There's only one problem, your friend just pointed out that the keyboard app has no autocorrect functionality. No suitable API exists, so you'll have to implement that yourself.



Your brother was adopted.
6:13p

What?!?! Why are you telling me this now?!
6:14p

*Accepted. He got in to SFU.
6:14p

Stupid autocorrect.
6:14p

In the domain of spelling correction, a measure known as Levenshtein distance is commonly used. The Levenshtein distance between a word v and another word u is the number of single letter modifications – adding, removing, or replacing a single letter – required to transform v into u For example, the Levenshtein distance between the word "adopted" and the word "accepted" is 3 because you transform the former into the latter by replacing the first 'd' and the 'o' with 'c's and by adding an 'e' after the last 'c'

Input

Input will consist of a single test case consisting of a dictionary and a number of test inputs to be autocorrected. The input starts with an integer w ($w < 10^5$) followed by w lines each of which contains a single word in lower-case letters only, describing the words in the dictionary. You can assume that no dictionary word will be specified more than once and every word will be less than 50 letters long. This is followed by a single line with the integer t ($t < 10^5$) and then t lines, each of which contains a single text input of lower-case letters only, less than 50 letters in length. Each of these inputs describes a user attempt to spell a word.

Output

For each of the t attempt inputs print a single line of output. If the attempted word occurs in the dictionary, print the word as output. Otherwise, print the list of words in the dictionary within Levenshtein distance 2 of the input in lexicographically increasing order. If no matching words exist in the dictionary, print a blank line.

Example

Input	Output
`8`	`cat`
`car`	
`cast`	`which witch with`
`cat`	
`cob`	
`comb`	
`which`	
`witch`	
`with`	
`3`	
`cat`	
`brm`	
`wich`	

Problem B. Bridge Master

Source file name:	bridgemaster.c, bridgemaster.cpp, bridgemaster.java, bridgemaster.py
Input:	`Standard`
Output:	`Standard`

A top-secret gathering of intellectual masterminds is occurring at SFU. These rather nerdy geniuses convene once in a while to compete against one another in a somewhat uncommon but highly skill-based activity, in which they may exercise their brains to tackle problems involving statistics and combinatorics. This is, of course, a meeting of the SFU bridge club.

N ($2 \leq N \leq 1000$, N is even) players, including yourself, are ready to play. The ith player aside from yourself has M_i ($1 \leq M \leq 1000$) masterpoints, a direct measure of their skill. The players are numbered in non-decreasing order of masterpoints - that is, $M_1 \leq M_2 \leq \ldots \leq M_N$. Meanwhile, you have K ($1 \leq K \leq 1000$) masterpoints.

The N players will be divided into $N/2$ pairs. Each pair's skill rating is the sum of the masterpoints that its two players have. You then play in a tournament, and receive a rank based on your skill rating. In particular, your rank is defined as the number of teams (including yours) whose skill rating is no less than your team's skill rating. Naturally, a lower rank is better (1 being the best rank).

You'd like to do as well as possible in this tournament, of course. The bad news is that you can't do anything about your current masterpoint count. The good news is that... well, let's just say you're on pretty good terms with the tournament organizer.

Given that the N players (including yourself) can be paired up however you'd like, what's the best (smallest) possible ranking you can achieve?

Input

The input begins with an integer T, the number of testcases. Each testcase begins with two integers N and K, the number of players and your personal number of masterpoints. This is followed by $N-1$ lines, which are the masterpoints of the other $N-1$ players (in non-decreasing order).

Output

For each test case, output the best possible ranking as an integer on its own line.

Example

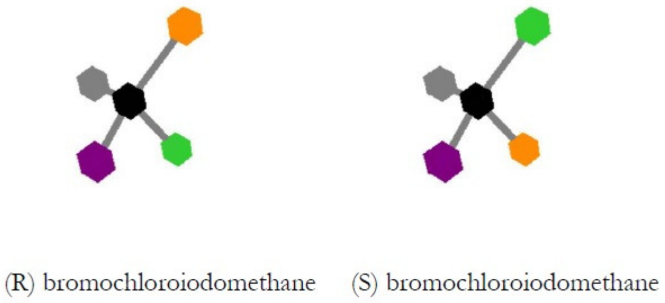
Input	Output

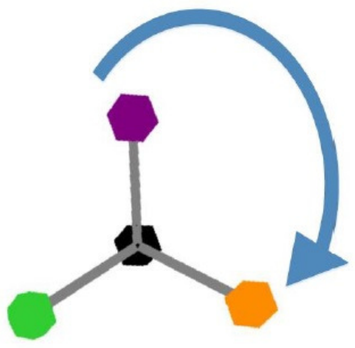
1
8 10
20
25
40
42
45
55
60
	2

Problem C. Chiral Center

Source file name:	chiralcenter.c, chiralcenter.cpp, chiralcenter.java, chiralcenter.py
Input:	`Standard`
Output:	`Standard`

We all know that atoms bond together to form molecules and that depending on the bonds that exist the same atoms can have very different properties. However, in some cases, molecules with the same atoms and bonds can have different properties if the atoms are arranged differently in three-dimensional space, a property known as chirality. The two molecules above both consist of the same four atoms bonded to a central carbon, but have different *chirality* and therefore different names.





Students at Simon Fraser University are first taught about chirality in CHEM 281. For this problem, we will only consider the existence of chiral centres when four different groups are bonded to a single carbon (atomic number 6), which always forms a tetrahedral shape around the central carbon. In CHEM 281, students are instructed to determine the chirality of a molecule by first ordering the groups bonded to the carbon by priority. For this problem, we will consider an atom to have a priority based on its atomic number (greater atomic number, higher priority). The students are then told to rotate the molecule so that they are looking through the central carbon towards the lowest priority group behind it, with the other groups forming a triangle around the carbon. Finally, students are told to observe the position of the highest priority group relative to the second highest priority group. If the second highest priority group is directly clockwise from the highest then the chiral centre is “Rectus”, if it is anti-clockwise the chiral centre is “Sinister.”

Input

For this problem, there will be at most 1000 test cases each of which will consist of 5 lines each with one integer n ($0 < n < 119$) followed by three double precision floating point values x, y, and z describing the atomic number and position of an atom, respectively ($-100 < x, y, z < 100$). In each case, one of the atoms will be a carbon ($n = 6$) and the other four will be other elements ($n \neq 6$). You are to assume the molecule has four bonds, one between each of the other atoms and the carbon. Input ends on EOF.

Output

For each test case, print a single line with either “Sinister”, “Rectus”, or “No chiral centre” to describe the chirality of the molecule in the test case.

Example

Input	Output

6 0 0 0
1 1 0 -0.70710678118
7 -1 0 -0.70710678118
17 0 1 0.70710678118
35 0 -1 0.70710678118
	Rectus

6 0 0 0
1 1 0 -0.70710678118
7 -1 0 -0.70710678118
35 0 1 0.70710678118
17 0 -1 0.70710678118
	Sinister

Problem D. Diminishing Marginal Value

Source file name:	diminishing.c, diminishing.cpp, diminishing.java, diminishing.py
Input:	`Standard`
Output:	`Standard`

To be a successful software engineer, you must have both strong technical skills and strong inter-personal skills. The relationship between achievement and these two skills is described by the formula

$$y = rp - |r - p|$$

where r is a person's level of technical skills, p is the level of interpersonal skills, and y is the amount of career success achieved by the software engineer.

Input

For this problem, you will determine which attribute an engineer should improve upon. Input will consist of a series of test cases, one per line, containing integers r and p ($0 \leq r, p \leq 10^6$). Input ends on EOF.

Output

For each test case, print a single line describing how the software engineer can best improve with one additional unit of skill. If their career success is increased most by an improvement in technical skills print "Technical", if their career success is improved most by an increase in interpersonal skills print "Interpersonal", otherwise print "Either".

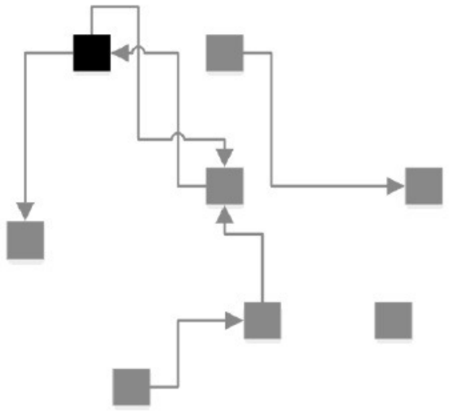
Example

Input	Output

10 5
20 50
0 0
	Interpersonal
	Technical
	Either

Problem E. Mark and Sweep

Source file name:	marksweep.c, marksweep.cpp, marksweep.java, marksweep.py
Input:	`Standard`
Output:	`Standard`



Computer software has access to two types of memory: static memory that is reserved at compile-time and *dynamic memory* that is allocated at runtime using an operator like `new` in `Java` or a system call like `calloc` in `C`. Since dynamic memory is allocated at runtime it may also be deallocated at runtime. In low-level languages this is done with a system call to a function like `free` in `C`. In high-level languages like Java, dynamic memory is deallocated automatically using a piece of software known as a "garbage collector."

Most garbage collectors use some variation of the mark and sweep algorithm. First, sections of static memory and all other memory blocks reachable through pointers contained within these blocks are marked for preservation. Next, those blocks of memory which have not been marked are deallocated. You will implement such a garbage collector.

Input

Input consists of a series of test cases, each test case begins with one line containing two integers b $(1 \leq b < 10^4)$ and s $(1 \leq s < b)$ indicating the number of blocks of memory and the number of static blocks, respectively. This will be followed by b lines each of which will contain three integers a $(1 \leq a < 2147483647)$, k $(1 \leq k < 2147483647)$, and p $(0 \leq p \leq b)$ describing the memory address of a block of memory, the block's length in bytes, and the number of pointers contained within a block. This will be followed by p integers on the same line indicating that the block contains a pointer to the block with each address. These integers will be either a valid block-address or 0 to indicate a null-pointer. No two blocks will have the same address and no block's address will be greater than another block's address while not also being greater than the sum of the block's address and its size. This will be followed by a single line of s unique integers corresponding to block addresses, indicating that these blocks are static.

Output

For each line of input, print a single line of output indicating the number of bytes of memory that can be deallocated by running the garbage collector.

Example

Input

```
8 1
268435465 4096 4 0 268435465 268455936 268484608
268455936 8192 0
268484608 1024 1 268435465
268496896 4096 1 268484608
268533760 8192 1 268496896
268570624 1024 0
268574720 32 1 268607488
268607488 1024 0
268435465
```

Output

```
14368
```

Input

```
1 1
1 1000 1 1
1
```

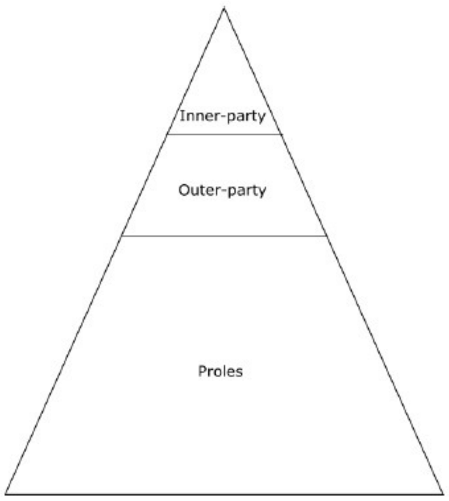
Output

```
0
```

Problem F. Our Duty to the Party

Source file name:	dutyparty.c, dutyparty.cpp, dutyparty.java, dutyparty.py
Input:	`Standard`
Output:	`Standard`

Comrade! Oceania is at war with Eurasia. Oceania has always been at war with Eurasia, but now the threat comes from within – members of our own outer-party. To prevent important secrets from being compromised the party practices strict compartmentalization. No single member of the outer-party knows anything important, but if a group of outer-party members were to piece together their knowledge they could severely damage the party. Fraternization between outer-party members is discouraged for this reason, but it is unavoidable. Given the heightened threat-level, the inner-party has determined that the outer-party must not be allowed full-knowledge of any important secret. For this reason, at least one outer-party member with knowledge of part of each secret must be disappeared.



Unfortunately, outer-party members, unlike the proles, are costly to replace – they have specialized skills. Worse still, when an outer-party member is disappeared, there is a strong chance that their close friends will defect to Eurasia. It is imperative that if an outer-party member is to disappear that all of their friends be disappeared as well. You are to design a computer program to plan disappearances to ensure that full knowledge of any party secret does not exist within the outer-party while preventing defections and minimizing human resources costs.

Input

Your program will handle a series of test cases, each of which will begin with a line containing three integers n (0 <n$\leq$ 5000) the number of party members, s ($0 < s < 11$) the number of secrets, and f ($0 < f < 20000$) the number of friendships. The next n lines will contain integer replacement costs for each of the party members as determined by the Ministry of Plenty. This will be followed by f lines, each containing two integers a and b, indicating that there is a bi-directional close-friendship between party members a and b. For compartmentalization reasons (you might be on the list!) party IDs have been anonymized and replaced with integers on the range $[0 \ldots n)$. Finally, the next s lines will each describe a party secret. Each of these lines will start with an integer q ($1 < q < n$) and be followed by q integers listing the anonymized IDs $[0 \ldots n)$ of the party members with knowledge of part of the secret. Input ends on EOF.

Output

For each test case, you should print a single line containing the total cost of ensuring that the outer-party cannot possibly derive any of the full secrets using the sum of its knowledge.

Example

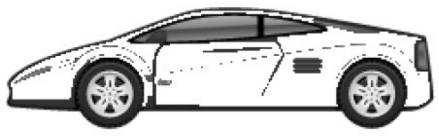
Input	Output

5 2 2
5
10
1
1
1
0 1
3 2
3 0 1 2
2 3 4
	2

Problem G. Profit Maximization

Source file name:	profit.c, profit.cpp, profit.java, profit.py
Input:	`Standard`
Output:	`Standard`

You've recently finished your business degree at Simon Fraser University and have been hired-on at the Vancouver office of a Chinese company Hainan Motors. The firm currently has no retail presence in the West, but plans are being made to market the 17Z sports car across Canada. Since Canadian consumers are currently unfamiliar with the marque, Hainan is still deciding whether it will market itself as an economy brand or a line of luxury vehicles. This decision will be made so as to maximize profit for Hainan Motors shareholders.



Consultants working in Canada have polled focus groups to determine whether or not a demographic group is likely to buy the 17Z within a specific price range and the engineering team in Haikou has provided an expected unit-price for the Canadian model. Using these data points and your computer skills, find the optimal retail price to ensure the largest profit for your employer.

Input

Input will consist of several test cases. Each test case will start with a line containing two integers, n ($2 \leq n \leq 10^4$) the number of demographic groups and m ($15000 < m < 40000$) the expected per-unit production cost of the 17Z. This will be followed by n lines, each with three integers q ($1 \leq q \leq 10^5$) representing the number of people in the demographic group, x representing the minimum price members of the demographic group will pay for the 17Z, and y representing the maximum price that members of the demographic group will pay for the 17Z ($0 \leq x < y \leq 500000$). Input ends on EOF.

Output

For each test case print a single line with the optimal retail price, if more than one retail price is optimal choose the highest; Hainan Motors would prefer to be seen as exclusive. An optimal price less than or equal to 500000 will always exist.

Example

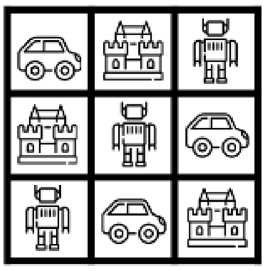
Hint: The minimum prices are important, luxury consumers buy cars partially for their exclusivity and will not purchase the car if it is too cheap.

Input	Output

5 17260
1000000 0 5000
10000 15000 21000
10000 25000 40000
5000 30000 45000
100 150000 400000
	40000

Problem H. Latin Squares

Source file name:	latinsquares.c, latinsquares.cpp, latinsquares.java, latinsquares.py
Input:	`Standard`
Output:	`Standard`



A Latin Square is an $n - by - n$ array filled with n different digits, each digit occurring exactly once in each row and once in each column. (The name "Latin Square" was inspired by the work of Leonhard Euler, who used Latin characters in his papers on the topic.)

A Latin Square is said to be in reduced form if both its top row and leftmost column are in their natural order. The natural order of a set of digits is by increasing value.

Your team is to write a program that will read an $n - by - n$ array, and determine whether it is a Latin Square, and if so, whether it is in reduced form.

Input

The first line of input contains a single integer n ($2 \leq n \leq 36$). Each of the n next lines contains n digits in base n, with the normal digits '`0`' through '`9`' for digits values below 10 and uppercase letters '`A`' through '`Z`' representing digit values `10` through `35`. All digits will be legal for base n; for instance, if n if `3`, the only legal characters in the n input lines describing the square will be '`0`', '`1`' and '`2`'.

Output

If the given array is not a Latin Square, print "`No`" on a single line (without quotation marks). If it a Latin Square, but not in reduced form, print "`Not reduced`" on a single line (Without quotation marks). If it is a Latin Square in reduced form, print "`Reduced`" on a single line (without quotation marks).

Example

Input	Output
3 012 120 201	Reduced
4 3210 0123 2301 1032	Not Reduced
11 0123458372A A9287346283 0285475A834 84738299A02 1947584037A 65848430002 038955873A8 947530200A8 93484721084 95539A92828 04553883568	No

Problem I. Halfway

Source file name:	halfway.c, halfway.cpp, halfway.java, halfway.py
Input:	`Standard`
Output:	`Standard`

A friend of yours has written a program that compares every pair of a list of items. With n items, it works as follows. First, it prints a `1`, and it compares item `1` to items `2, 3, 4, ..., n`. It then prints a `2`, and compares items `2` to items `3, 4, 5, ..., n`. It continues like that until every pair of items has been compared exactly once. If it compares item x to item y, it will not later compare item y to item x. Also, it does not compare any item to itself.

Your friend wants to know when his program is *halfway done*. For a program that makes an odd number of total comparisons, this is when it is doing the middle comparison. For a program that makes an even number of total comparisons, this is when it is doing the first of the two middle comparisons.

What will the last number printed be when the program is halfway done?

Note that since the earlier items have more comparisons than the later items, the answer is not simply $n/2$.

Input

The input consist of a single line containing the integer n ($2 \leq n \leq 10^9$).

Output

Print, on a single line, the last number your friend's program prints when it is halfway done.

Example

Input	Output
4	1
7	2
10	3
1919	562
290976843	85225144

Problem J. Purple Rain

Source file name:	purplerain.c, purplerain.cpp, purplerain.java, purplerain.py
Input:	`Standard`
Output:	`Standard`

Purple rain falls in the magic kingdom of Linearland which is a straight, this peninsula.

On close observation however, Professor Nelson Rogers finds that the purple rain is actually a mix of red and blue raindrops.

In his zeal, he records the location and color of the raindrops in different locations along the peninsula. Looking at the data, Professor Rogers want to know which part of Linearland had the "least" purple rain.

After some thought, he decides to model this problem as follows. Divide the peninsula into n sections and number the west to east from 1 to n. Then, describe the raindrops as a sequence of **R** and **B**, depending on whether the rainfall in each section is primarily red or blue. Finally, find a sub sequence of contiguous sections where the difference between the number of **R** and the number of **B** is maximized.

Input

The input consist of a single line of containing a string of n characters ($1 \leq n \leq 10^5$), describing the color of the raindrops in section **1** to n.

It is guaranteed that the string consist of uppercase **ASCII** letters '**R**' and '**B**' only.

Output

Print, on a single line, two space-separated integers that describe the starting and ending positions of the part of Linearland that had the least purple rain. These two numbers should describes an inclusive range; both numbers you print describe sections included in the range.

If there are multiple possible answers, print the one that has the westernmost starting section. If there are multiple answers with the same westernmost starting section, print the one with the westernmost ending section.

Example

Input	Output
`BBRRBRRBRB`	`3 7`
`BBRBBRRB`	`1 5`

Problem K. Unloaded Dice

Source file name:	unloaded.c, unloaded.cpp, unloaded.java, unloaded.py
Input:	`Standard`
Output:	`Standard`

Consider a six-sided dice, with sides labeled 1 through 6. We say the dice is *fair* if each of its sides are equally likely to be face up after a roll. We say the dice is *loaded* if it isn't fair. For example, if the side marked is 6 is twice to come up as than any other side, we are dealing with a loaded dice.

For any dice, define the *expected result* of rolling the dice to be equal to the average of the values of the sides, weighted by the probability of those sides coming up. For example, all six sides of a fair dice are equally likely to come up, and this the expected result of rolling it is $(1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$.

You are given a loaded dice, and you would like to *unload* it to make it more closely resemble a fair dice. To do so, you can erase the number on one of the sides, an replace it with a new number which does not need to be an integer or even positive. You want to do so in such a way that.

- The expected result of rolling the dice is 3.5, just like a fair dice.
- The difference between the old label and the new label on the side you change as small as possible.

Input

The input consist of a single line containing six space-separated non negative real numbers $v_1 ... v_6$, where v_i represents the probability that side i (currently labeled by the number i) is rolled.

It is guaranteed that the given numbers will sum to 1.

Output

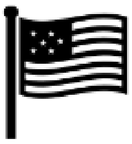
Print, on a single line, the absolute value of the difference between the new label and old label, rounded and displayed to exactly three decimal places.

Example

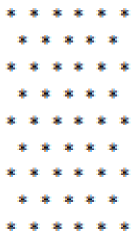
Input	Output
`0.16666 0.16667 0.16667 0.16666 0.16667 0.16667`	`0.000`
`0.2 0.2 0.1 0.2 0.2 0.1`	`1.000`

Problem L. Star Arragements

Source file name: arrangements.c, arrangements.cpp, arrangements.java, arrangements.py
Input: `Standard`
Output: `Standard`



The recent vote in Puerto Rico favoring United Stared statehood has made flag makers very excited. An updated flag with 51 stars rather than the current one with 50 would cause a huge jump in U.S.flag sales. The current patter for 50 stars in five rows of 6 stars, interlaced with four offset rows of 5 stars. The rows alternate until all stars are represented.



This pattern has the property that adjacent rows differ by no more than one star. We represent this star arrangement compactly by the number of stars in the first two rows: **6,5**.

A 51-star flag that has the same property can have three rows of 9 stars, interlaced with three rows of 8 stars (with a compact representation of 9,8). Conversely, if a state were to leave the union, one appealing representation would be seven rows of seven stars (7, 7).

A flag pattern is *visually appealing* if it satisfies the following conditions:

- Every other row has the same number of stars.
- Adjacent rows differ by no more than one star.
- The first row cannot have fewer stars than the second row.

Your team sees beyond the short-term change to 51 for the U.S. flag. You want to corner the market on flags for any union of three or more states. Given the number S of stars to draw on a flag, find all possible *visually appealing* flag patterns.

Input

The input consist of a single line containing the integer S ($3 \leq S \leq 32,767$).

Output

On the first line, print S, followed by a colon. Then, for each visually appealing flag of S stars, print its compact representation, one per line.

This list of compact representations should be printed in increasing order of the number of stars in the first row; if there are ties, print them in order of the number of stars in the second row. The case 1-by-S ans S-by-1 are trivial, so do not print those arrangements.

The compact representations must be printed in the form “**x,y**” with exactly one comma between **x** and **y** and no other characters.

Example

Input	Output
3	3:
	2,1
50	50:
	2,1
	2,2
	3,2
	5,4
	5,5
	6,5
	10,10
	13,12
	17,16
	25,25
51	51:
	2,1
	3,3
	9,8
	17,17
	26,25

Problem M. Forbidden Zero

Source file name:	forbidden.c, forbidden.cpp, forbidden.java, forbidden.py
Input:	`Standard`
Output:	`Standard`

You're writing the positive integers in increasing order starting from one. But you've never learned the digits zero, so you omit any number that contains a zero in any position. The first ten integers you write are: 1, 2, 3, 4, 5, 6, 7, 8, 9 and 11.

You have just written down the integer n (which is guaranteed to not contain the digit zero). What will be the next integer that you write down?

Input

The input consist of a single line containing the integer n ($1 \leq n < 10^6$). It is guaranteed that n does not contain the digit zero.

Output

Print, on a single line, the next integer you will be writing down.

Example

Input	Output
99	111
1234	1235