Problem A. Apollo's Daughter

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Phemonoe, the first oracle of Delphi, possibly the daughter of Apollo himself, is in a series of n games. The games happen sequentially. You cannot go back to play in a previous game that is already done, as time travel is not in the realm of Apollo's powers. She can choose not to participate in a game though. In these games, she can place bets on each of them and receive money that is some multiple of the bet she placed. Usually you shouldn't know the result of each game. But since she is an oracle of Delphi, Apollo already told her what would happen exactly. However, Apollo did warn her to not be too greedy. Being too suspicious and win too much is certainly not a good idea. Zeus might not be so happy if he finds out what Apollo did, you see.

Unfortunately, after a few glasses of good quality Greek wine, Phemonoe's judgment is not be as good as she needs it to be. As her confidence grows, she also starts to raise her bets. As her adviser, your job is to maximize her profit. Note that she will not listen to you in terms of the total number of games she wants to play and the amount she is going to bet, but she will listen to your advice as to which games to participate.

Input

The first line of the input will contain a single integer representing the total number of games n ($1 \leq n \leq 5 \cdot 10^4$). On the second line, there are n integers $p_1, p_2, \ldots, p_n$, representing the multiplier for each game. Phemonoe's winning of each game will be the product of her bet and the multiplier. Negative multipliers mean losing. For every $i \in \{1, \ldots, n\}$, you may assume $|p_i| \leq 5 \cdot 10^4$.

The third line contains two non-negative integers a and b ($0 \leq a, b \leq 100$), indicating that Phemonoe will bet $j^2 + aj + b$ where $a^2 \geq 4b$ on the j-th game she participates.

Output

Print n integers on a line, representing the maximum profit if Phemonoe decided to participate in $1, 2, \ldots, n$ games respectively.

Example

standard input	standard output
3 1 2 3 0 0	3 14 36
5 1 -1 1 -1 1 3 2	6 18 38 44 26

Problem B. Boolean Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The boolean game starts with boolean formula written on the board. And then, Alice and Bob will take turns in deciding the value of one variable (true or false). Once the value of every variable is decided, if the formula is satisfied, Alice wins. Otherwise, Bob will win.

Check if Alice can win this game if she pick first, assuming that both Alice and Bob are playing optimally.

Input

The first line contains an integer n ($1 \leq n \leq 10$) indicating the number of variables in the boolean formula. The i-th variable will be represented by the i-th English alphabet in upper cases. That is, the first variable is A, and the third variable is C and so on.

The boolean formula will be given in its second line which has length no more than 256 characters. A boolean formula must be in one of the following five forms.

- `var`: `var` is a variable.
- `( formula1 )`: `formula1` is a boolean formula.
- `not formula1`: `formula1` is a boolean formula.
- `formula1 or formula2`: both of `formula1` and `formula2` are boolean formulas.
- `formula1` and `formula2`: both of `formula1` and `formula2` are boolean formulas.

You may assume that there are blanks to separate the variables and operators in the input file.

There are four kinds of boolean operators: `and`, `or`, `not`, and brackets `()`. Note that the first three kinds of operator are in lower cases. The operator precedence given from high to low as follows: `()` ¿ `not` ¿ `and` ¿ `or`.

Output

Output the winner's name ("`Alice`" or "`Bob`").

Example

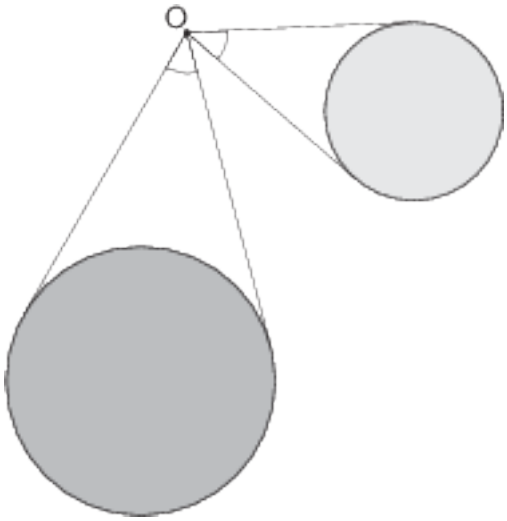
standard input	standard output
1 A and not A	Bob
1 A or not A	Alice
3 (A or C) and B	Alice

Problem C. Choose the Point

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given n circles on the plane. Your task is to determine if the point with the next properties exists:

- Point must lie strictly outside of any circle.
- Angle of view for all n circles must be the same, i.e. angle between two tangents from this point to each circle must be the same for all n circles.
- Tangent from this point to some circle cannot intersect the other circle.



Input

First line of the input consists an integer n ($2 \le n \le 4$) — number of the circles. Each of the next n lines contains three integers x_i, y_i, z_i ($-100 \le x_i, y_i \le 100$, $1 \le r_i \le 100$). You may assume that no two circles have the common point.

Output

If it is impossible to find such a point, print "NO". Otherwise print "YES" in the first line of the output file and two real numbers x_o and y_o in the second line. Difference between maximum and minimum angle of view for all circles must not be greater than 10^{-4}.

Examples

standard input	standard output
2 2 2 1 5 2 1	YES 3.5 0.0
3 0 4 1 4 4 1 4 0 1	YES 2.0 2.0
3 0 4 1 4 4 2 4 0 1	NO

Problem D. Darkshire

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Stitches is the terror of Darkshire. He emits a putrid bile along his path that is stinky, disgusting and slows you by 35% percent when walking in it. So in short, you don't want to stand in them. With Stitches's movement, he can potentially cut the map into several separated areas. Two areas are separated if there is no way one can reach the other without crossing Stitches's bile. Stitches always walks along a single direction (up, down, left or right) for a unit of length and then decides whether to change his direction or not. In order to bring peace to Darkshire, the heroes will kill him as soon as possible. You may assume that Stitches only emits bile for at most 2048 units of length, since the heroes will not allow him to walk more.

Unfortunately, Stitches' bile does not disappear after his death. As the mayor of Darkshire, you want to determine how many separated areas there are after Stitches walks through.

Input

Input consists of one line containing the sequence of Stitches' movement as a string composed with '`U`' for up, '`D`' for down, '`L`' for left and '`R`' for right. The length of the string will not exceed 2048.

Output

Print the number of separated areas on one line.

Example

standard input	standard output
LURD	2
LUDR	1
LDRDRURRDLUURURD	2
LLUURRDDRULLLLUR	5

Problem E. Encryption

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The Gödel numbering system works in next way. It assigns a number to each symbol, in this case we use capital alphabets (`A` to `Z`) and assign them to natural numbers 1 to 26. So that `A` is assigned to 1, `B` is assigned to 2 and so on. So a word such as `CAMP` will become a sequence $(3, 1, 13, 15)$. Then, we represent it as product of prime powers. A sequence $(a_1, a_2, \ldots, a_n)$ should be encoded into $2^{a_1} \cdot 3^{a_2} \cdot \cdots \cdot p_n^{a+n}$ where p_i is the i-th prime number. I.e., the Gödel number of `CAMP` is $2^3 \cdot 3^1 \cdot 5^{13} \cdot 7^{15}$.

The Gödel number of a long word might be very large. Your friend Alice would like to leave an encrypted message for you by using Gödel numbers, but she is lazy to write all the digits. Alice uses a triple (ℓ, r, p) to represent a Gödel number g where ℓ is the length of the corresponding word, p is a prime, and $r \equiv g \pmod{p}$. However, Alice doesn't notice that two words can generate the same triple. For instance, the Gödel numbers `EA` and `JA` are 96 and 3072 respectively, but Alice may use $(2, 3, 31)$ to represent both of them. Let's try to decode Alice's message!

Input

The input will be given on a line containing three integers. The first integer ℓ ($1 \le \ell \le 8$) is the length of this word. The second integer r represents the remainder of the result. The third integer is the prime number p used in modulo operation. We guarantee that $0 \le r < p < 2^{31}$.

Output

Output the word that generates this triple if it is unique. Output "`ambiguous`" if more than one possibility exists. Output "`not a word`" if there exists no word that can produce this triple.

Example

standard input	standard output
1 2 3	ambiguous
3 11385 13339	COW
7 123123123 2089141547	ambiguous
7 0 2089141547	not a word

Problem F. Fire

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

There are some fiery buildings in a city and fire fighters are trying to control the fire. But the speed of propagation of fire is more than what can be controlled. Fire fighters discovered that if they only extinguish the boundary of a fire zone, it can be controlled better. To do this, they must know how the fire spreads. Here is the description of fire propagation process and structure of buildings:

- A fiery building ignites its neighbors after T steps. T is $\lceil(10 \cdot a/b)\rceil$, where a is the area of non-fired building and b is the area of ignited one.
- Two buildings are neighbors if the smallest distance among all pairs of their points is less than 30.
- Each building is a non-self-intersecting polygon.

Buildings neither overlap nor touch each other.

Input

Input starts with an integer m ($1 < M \leq 100$), the number of buildings. After that, there are M lines each containing an integer $3 \leq L \leq 33$ followed by L pairs of integers X_i and Y_i ($-1000 \leq X_i, Y_i \leq 1000$) that are coordinates of vertices of i-th building (either in clockwise or counterclockwise direction). After that, there is an integer $0 \leq P < M$ representing that the P-th building is ignited in the zero cycle).

Then there is an integer C ($0 \leq C \leq 300$) — number of the cycles.

Output

Print the indices of ignited buildings in cycle C in increasing order. It should be noted that two floating point numbers a and b are assumed equal if their absolute difference is less than 10^{-8}.

Example

standard input	standard output
4 4 1 1 1 2 2 2 2 1 3 3 3 4 3 4 4 4 22 1 23 1 23 2 22 2 3 34 3 34 4 35 4 0 7	0 1

Problem G. Goto

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

After reading the famous Dijkstra's book, Bill decided to get rid of all the goto statements in his code. The only kind of flow control statements in Bill's source code was the `if-goto`. To decrease the goto-density, Bill have to try and eliminate all the `if-goto` statements in his source code written in a C-like language and replace them with `do-while` loops in the following manner.

- Assume the `if-goto` statement looks like "`if (boolean_expression) goto some_label;`".
- Insert a copy of "`do {`" right after where `some_label` is declared.
- Replace '`if`' by '`} while`'.
- Remove '`goto some_label`'.

For example, the following code

```
int main() {
    int score;
    get_score:
    scanf("%d",&score);
    if (score < 0 || score > 100) goto get_score;
    if (score < 60) goto fail;
    fail:
    puts("you are failed!");
    return 0;
}
```

will be modified into

```
int main() {
    int score;
    get_score: do {
    scanf("%d",&score);
    } while (score < 0 || score > 100) ;
    } while (score < 60) ;
    fail: do {
    puts("you are failed!");
    return 0;
}
```

It is not too surprising that the code above cannot be compiled. Here is your new task: given a sequences of statements, please determine whether all `if-goto` statements can be replaced by `do-while` loops without changing the output of your code. For simplicity, you may assume all statements are in one of the following two forms.

- Form 1: "`line_`x`: puts("`x`");`" where x is corresponding line number of this statement and "`puts("`x`");`" prints the line number x.
- Form 2: "`if (expr_`x`()) goto line_`y`;`" where x is the corresponding line number of this statement and `line_`y is a valid label in the program. "`expr_`x`()`" will return **true** on first x invocations, and it will return **false** afterward.

Note that if the modification makes the code unable to be compiled, then the output of the program should be considered different from the original output.

Input

The input consists of a sequence of statements. Each statements will be on a single line, which starts on line 1. There can be three kinds of lines: statements in form 1, statements in form 2, and "`END`". "`END`" will indicate the end of a test case. It can only be the last line of a test case, and it should not be considered as a part of the program. There are at most 10^4 statements in a single test (`END`'s are excluded).

Output

Print "`good`" on a single line if replacing all `if-goto` statements with `do-while` will not change the output of the program. Otherwise print "`bad`". Note: you should output "`bad`" if the program becomes no longer compilable.

Example

standard input	standard output

`line_1: puts("1");`
`if (expr_2()) goto line_1;`
`END`
	`good`

`line_1: puts("1");`
`line_2: puts("2");`
`line_3: puts("3");`
`if (expr_4()) goto line_2;`
`line_5: puts("5");`
`if (expr_6()) goto line_2;`
`END`
	`good`

`line_1: puts("1");`
`if (expr_2()) goto line_5;`
`line_3: puts("3");`
`line_4: puts("4");`
`line_5: puts("5");`
`END`
	`bad`

Problem H. Halting Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A *Turing Machine* has a certain number of states, a tape of infinite length which admits writing symbols from a certain set, and runs within a certain set of transition rules. Starting from an initial state with a working head that points at a certain location of the tape, it can decide (according to the transition rules which account the current state and the current symbol the head is pointing at on the tape) either to move its head towards certain direction or to halt, and also which symbol it wants to write back to the tape.

For several inputs to the Turing Machine check if it will halt within 10 steps, that is, if it will enter the special "halting state" within this amount of steps.

Input

First the description of a Turing Machine will be given. It starts with an integer n ($1 \le n \le 10$) representing the number of states of this Turing Machine on a separate line. State 1 is the starting state, and state n is the only halting state, i.e., the Turung Machine will halt after entering it.

The next $n-1$ lines will be the transition rules of state 1 to $n-1$ (note that since state n is the halting state, it does not have any transition rules). To simplify the problem, there will only be three symbols used in the Turing Machines of this problem, namely 0, 1 and 2. Each line of the transition rule contains 3 tuples in the form of (x, y, z) separated by a blank, in which x represents the next state the Turing Machine will transition to, y represents the direction the head is going (1 for right and -1 for left), and z represents what should be written back onto the tape. Note that the head will first write back the symbol, then move. For line i in these $n-1$ lines, the three 3-tuples will represent the transition of state i when it reads symbol 0, 1 and 2 respectively. For instance, the second 3-tuple of the fifth line will represent the transition rule of state 5 if the head is pointing at a symbol 1.

After that an integer m ($m \le 100$) indicating the number of queries is given on a separate line. Each query will be written on a single line in which the first number x ($0 \le x \le 10$) is the number of symbols in this input, after that x symbols will follow. The symbols will be written one after the other on the tape with the initial head position pointing at the first symbol. There will only be symbols 0 and 1 in the input. Symbol 2 is representing a blank symbol in the Turing Machine. You may assume that every symbol on the tape before the input and after the input is 2. For instance, an input `3 1 0 0` will actually look like this on the tape: "`... 2 2 2 1 0 0 2 2 2 ...`", where the head will point at the symbol 1 initially. The symbol 2 will extend infinitely on both ends.

Output

For each query, print "`yes`" on a line if the machine does halt in 10 steps and "`no`" if not.

Example

standard input	standard output
3 (2, 1, 0) (2, 1, 1) (2, 1, 2) (3, 1, 0) (1, -1, 0) (1, -1, 2) 3 3 1 0 0 2 1 1 1 1	yes yes no
4 (2, 1, 1) (3, 1, 0) (1, -1, 2) (1, -1, 0) (1, -1, 1) (1, -1, 2) (2, 1, 1) (2, 1, 0) (4, -1, 2) 2 3 0 0 0 5 0 0 0 0 0	yes no
2 (1, -1, 2) (2, -1, 0) (1, 1, 2) 2 2 0 0 1 1	no yes

Problem I. Inverse the MTF

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The Move-to-Front (MTF) transform is an encoding scheme which maps the input data into a sequence of numbers. Entropy encoding schemes often achieve better compression ratio on the data encoded by the MTF transform. The MTF transform is quite simple. The following scheme is the MTF transform on string consisting of only characters in lowercase.

1. Maintain a list of characters in lowercase. Initially, the list is sorted in the lexicographic order. I.e., the list is `[abcdefghijklmnopqrstuvwxyz]` at the beginning.
2. Read a character α from the string. Output the index of α in the list, then move α to the front of the list.
3. Repeat the previous step until all characters in the string are read.

For example, the following is how the transform above works on the string "`icpc`".

1. The first character '`i`' has index 8 in `[abcdefghijklmnopqrstuvwxyz]`. Output 8, then move '`i`' to the front of the list.
2. The second character '`c`' has index 3 in `[iabcdefghjklmnopqrstuvwxyz]`. Output 3, then move '`c`' to the front of the list.
3. The third character '`p`' has index 15 in `[ciabdefghjklmnopqrstuvwxyz]`. Output 15, then move '`p`' to the front of the list.
4. The fourth character '`c`' has index 1 in `[pciabdefghjklmnoqrstuvwxyz]`. Output 1, then move '`c`' to the front of the list.

The MTF transform maps "`icpc`" into the sequence (8, 3, 15, 1). Please write a program to inverse the MTF transform. In other words, your program reads a sequence of numbers $(a_1, \ldots, a_n)$, then compute the string s such that the MTF transform maps s into $(a_1, \ldots, a_n)$.

Input

Input consists of two lines. The first one contains a positive integer n, $(1 \leq n \leq 100)$, which indicates the length of the sequence. The second line contains n integers $(a_1, \ldots, a_n)$, separated by blanks. The input sequence is $(a_1, \ldots, a_n)$, and $a_i \in \{0, 1, \ldots, 25\}$ for $i \in \{1, \ldots, n\}$.

Output

Print the string s such that MTF transform maps s into $(a_1, \ldots, a_n)$.

Examples

standard input	standard output
4 8 3 15 1	icpc
6 1 1 13 1 1 1	banana

Problem J. Jackpot

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The Byteland National Lottery works in the following way. The lottery consists of rounds, each lottery ticket plays only in the round the ticket is assigned for.

Each citizen of Byteland can buy no more than one ticket for the current round. When he buys his first lottery ticket, he become addicted and starts to buy a ticket for each subsequent round until he wins the jackpot.

Rules of the lottery are as follows:

- At the beginning (before Round 1), the bank is empty.
- After the each round all money earned from the selling the tickets assigned for this round are added into the bank.
- If the amount of money in the bank is enough to pay the jackpot, then the owner of one of the tickets for this round wins the jackpot (winner is chosen equiprobably between those tickets) and the bank becomes empty regardless of the rest of the money in the bank after the jackpot was paid.
- If the amount of money in the bank is not enough to pay the jackpot, owner of each ticket for this round gets the partial reimbursement from the bank — half of the price of ticket.

Given that N citizens bought atlest one ticket, price for one ticket is L bytalers, jackpot is equal to J bytalers. For each citizen calculate the expected amount of money he will spend until he wins the jackpot.

Input

First line of the input contains three integers — N ($1 \le N \le 100\,000$), L ($2 \le L \le 1\,000\,000\,000$, L is even), K ($1 \le J \le 1\,000\,000\,000$).

Second line of the input contains N integers d_i ($1 \le d_i \le 1\,000\,000\,000$), where d_i is the round number when i-th citizen started to play the lottery.

Output

Print N lines. i-th of those lines must contain the expected amount of money spent until winning the jackpot for the i-th citizen with absolute or relative error 10^{-9} or better.

Example

standard input	standard output
3 2 11 1 3 1	12.66666666667 10.66666666667 12.66666666667

Problem K. Keyboard Typing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Kenny's typing speed is not fast, however, it is very steady. He strokes the keyboard exact once per second, and he always presses the keys correctly. Without interfering, Kenny has to spend n seconds to type a string s of n characters. In order to reduce the time spent for typing, Kenny copied a string b to the clipboard. Therefore, he can use the paste function to input many characters. Assume that pasting b also takes only one keyboard stroke for Kenny. If Kenny copied "`barcel`" before typing "`barcelona`", then Kenny can finish it in 4 seconds: pasting "`barcel`", then pressing '`o`', then pressing '`n`' and then pressing '`a`'. Please write a program to compute the minimum time required for Kenny to type a string s when he copied string b to the clipboard.

Input

Input consists of exactly one line containing two strings s and b separated by blanks. Kenny is going to type s with b copied to the clipboard. The length of s is at most 10^4, and the length of b is at most 100.

Output

Print the minimum time (in seconds) for Kenny to type string s with string b copied to the clipboard.

Example

standard input	standard output
barcelona barcel	4
abaceba ba	5

Problem L. Length of Period

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	256 mebibytes

For a string $A = a_1 a_2 \dots a_n$ the *length of period* of A is the smallest integer $1 \le p \le n$ such as $a_{i+p} = a_i$ for all $1 \le i \le n - p$.

Given the string S, you must process queries "find the length of period of the substring of S between L-th and R-th position, inclusively".

Input

The first line of the input contains a non-empty string S of at most 10^5 lowercase English letters. Next line contains one integer Q ($1 \le Q \le 10^5$) — the number of queries. Each of the next Q lines contains one query — two integers L and R ($1 \le L \le R \le |S|$).

Output

For each query print one integer — answer to this query.

Example

standard input	standard output
xxxyxxx 3 1 3 1 4 2 5	1 4 3
xyzxyzxyz 1 1 9	3