Problem A. Russian Dolls

Input file:	`a.in`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Maybe you know the famous russian souvenir Russian Dolls. It looks like a set of nested wooden dolls. A doll with a smaller size is placed inside a bigger one. Let's consider all dolls taken apart. Each doll has an outer volume out_i which is the volume it occupies in space and an inner volume in_i — the volume of the empty space inside the doll. You may assume that you can put one doll inside another if the outer volume of the first doll is strictly less than the inner volume of the second one. If two or more dolls are inside another one they can't lie one near the other, they must be nested.

For each doll the cost of unit of empty space — $cost_i$ is known. You must pay exactly $cost_i$ for each unit of empty space which directly belongs to the i-th doll (but not to ones inside it). You may arrange the dolls the way you want as long as you are not contradicting the rules. The objective is to find an arrangement of nesting the some dolls such that the overall cost you have to pay is minimized.

Input

First line contains an integer N ($1 \le N \le 1000$) which is the number of dolls you have. The i-th of the next N lines contains three integers out_i, in_i, $cost_i$ ($1 \le in_i < out_i \le 1000$, $1 \le cost_i \le 1000$), which are the outer volume, inner volume and the empty space cost of the i-th doll.

Output

Single integer P which is the minimum possible cost you should pay.

Examples

a.in	standard output

3
5 4 1
4 2 2
3 2 1
	7

Problem B. Intercity

Input file:	`b.in`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

A couple of years ago the Ukrainian Railway System seemed to be very convenient. For any two cities there was a direct train (don't be confused with "directed"), which travelled between them. Anyone had to pay only B UAH (the local currency) to get from his current place to the desired destination.

But not long ago great changes happened in Ukraine. A lot of new trains have been launched. Each of the new trains replaced an old one and the cost of its journey has been established to A UAH. So there is a single direct train (new or old) which still runs between any pair of cities now. Each train runs in both directions and the cost of the journey doesn't depend on it.

There are N large cities in Ukraine and you live in the city number 1. You want to get to the city number N, and you look for the cheapest way of going there, regardless of the number of transfers.

Input

The first line contains four integers N, K, A and B ($2 \le N \le 5 \cdot 10^5$, $0 \le K \le 5 \cdot 10^5$, $1 \le A, B \le 5 \cdot 10^5$), which are the number of cities, the number of new trains, the new cost of the journey and the old cost of the journey. K lines follow. Each of them contains two integers u_i and v_i ($1 \le u_i, v_i \le N$) which means that a new train is launched between city u_i and city v_i. u_i and v_i don't coincide. Every pair of cities appears at most once.

Output

Single integer P which is the cost of the cheapest way to go from city 1 to city N.

Examples

b.in	standard output

5 4 1 4
1 2
2 3
2 4
3 5
	3

Problem C. Count (Binary Input!)

Input file:	`c.in`
Output file:	`standard output`
Time limit:	2 seconds (3 seconds for Java!)
Memory limit:	512 mebibytes

You have:

A matrix of positive integers, with the property that all rows and all columns are sorted in ascending order (i.e. $A[i,j] \geq A[i-1,j]$ and $A[i,j] \geq A[i,j-1]$ for all i, j).

One or several pairs of numbers (X,Y) with the property that $Y \geq X$. For each (X,Y) pair, count how many numbers from the matrix are greater than or equal to X but smaller than or equal to Y.

Input

The input file is a **binary** file containing signed 32-bit integers (in little-endian format). The input file consists of:

- One integer N representing the number of rows (no more than 10^4).
- One integer M representing the number of columns (no more than 10^4).
- $N \times M$ positive integers, representing the values from the matrix, row by row
- An unspecified number of integers, representing the (X,Y) pairs, one pair at a time. There will be at least one pair and at most 100 pairs in the file — and there will not be an incomplete pair at the end of the file.

Output

For each pair you should write to standard output a value representing how many numbers in the matrix are greater than or equal to X but smaller than or equal to Y.

Examples

c.in	standard output

`2 4`
`1 5 10 10`
`2 10 20 99`
`10 99`
`2 9`
`100 1000`
`10 10`
	`5`
	`2`
	`0`
	`3`

Note

In the sample tests are not in binary form (for obvious reasons). Real file, corresponding to sample tests, is binary file with 18 32-bit integers (i.e size of file is equal to $18 \cdot 4 = 72$).

Problem D. Chess (Division 1 Only!)

Input file:	`d.in`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

There are two white rooks and one black king on a chess board. White starts. You're playing for White and need to checkmate the black king as soon as possible.

Write a program that will determine the minimum number of moves White needs to perform to checkmate black king, assuming Black follows the best possible strategy for prolonging the game.

Some information about the chess rules:

1. The position described above is not legal in chess since there is no white king on the board. Apart from that the game is according to the chess rules.
2. The board size is 8×8 cells. Columns of the board are labeled by the letters a to h, and the rows by the digits 1 to 8.
3. The players (White and Black) alternately move one piece of their own color at a time. In the context of this problem we're only counting moves made by White.
4. The rook moves horizontally or vertically, through any number of unoccupied squares. It cannot jump over or stay in the same cell as another piece of the same color, namely the other white rook.
5. A check is a threat to capture the king on the next move turn, i.e. a position in which one of the rooks is on the same line as the king.
6. A king can move one square in any direction (horizontally, vertically, or diagonally) unless the move would place the king in check. If this condition holds, it's not prohibited for king to take over a cell already occupied by one of the rooks. In this case the rook is removed from play altogether.
7. Checkmate is a position in which a king is threatened with capture (i.e. is in check) and there is no legal move to escape the threat.
8. Stalemate is a situation where the player whose turn it is to move is not in check but has no legal move. The rules of chess provide that when stalemate occurs, the game ends as a draw (which is not acceptable for White).

Input

The first line of the input contains the number of positions T ($1 \leq T \leq 10^5$), and each of the following lines contain the description of a single position. All positions in the input are different.

A position is represented by the location of the three pieces on the board: the black king first and then the two white rooks. It's guaranteed that no two pieces share a cell and there is no check at the initial position.

Output

Output one line for each position in the input. The line should contain the minimum number of moves which White needs to make to guarantee the checkmate, starting from the corresponding position. In case it's not possible to reach the goal, output 0 for relevant position.

Examples

d.in	standard output
2 c7 f1 g6 h6 c3 g8	2 1

Problem E. A+B (Division 1 Only!)

Input file:	`e.in`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

There is a computer, which has two memory cells (let us denote these cells by the letters a and b). Each cell (variable) stores some integer at any time. The computer can execute only two instructions $a+=b$ and $b+=a$. The first instruction increases the value of the variable a by the value stored in the variable b. The second one, respectively, increases the value of b by the value a. A program for this computer consists of a sequence (possible empty) of such instructions. The instructions are executed in the appropriate order. Your task is to determine whether the given value S can be obtained in some cell after executing some program.

Input

The input file contains three integers: the initial value of the variable a, the initial value of the variable b and the required value S ($0 \le a, b, S \le 10^{18}$).

Output

Output "`YES`" if the required value can be obtained as a result of some program execution, or "`NO`" otherwise.

Examples

e.in	standard output
1 2 3	YES
3 4 5	NO
3 4 17	YES

Problem F. Dragon Pattern (Division 1 Only!)

Input file:	`f.in`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

The *dragon curve* is a polygonal chain with segments of length 1, defined recursively. We choose some point on the plane and one of four directions, collinear to the coordinate axes. The left(right) dragon of order n is constructed in the following way:

- if n is equal to 0, we construct a segment of length 1 from the current point in the current direction, moving to endpoint;
- otherwise we construct the left dragon of order $n-1$ from the current point in the current direction, turn 90 degrees to the left(right) in the endpoint, and construct the right dragon of order $n-1$;

Let us construct the left dragon of order n, starting from the origin (point $(0,0)$) in the positive direction of the axis OX.

Given some pattern as a sequence of directions, in which sequential movements by unit length is performed, your task is to determine how many times does the given pattern occurs in the constructed dragon curve.

Input

Input file contains integer n ($0 \le n \le 10^9$) and string S consisting of symbols '`R`', '`L`', '`U`' and '`D`', which defines the pattern. '`R`' denotes rightward movement (positive direction of the axis OX), '`L`' denotes leftward movement (negative direction of the axis OX), '`U`' denotes upward movement (positive direction of the axis OY), '`D`' denotes downward movement (negative direction of the axis OY). The length of the string S is in the range from 1 to 10^6.

Output

Output number of given patterns in the left dragon of the order n, constructed in the positive direction of an axis OX, modulo 10^9+7.

Examples

f.in	standard output
2 RU	1
3 L	3
4 LDL	2

Problem G. Points

Input file:	`g.in`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

John is a fan of amusement parks. He goes every weekend and plays different games. This weekend he found a challenging one — it is a target shooting game. The targets are placed along a straight line. For all target positions i (assume the target numbering goes from right to left), there are three possible points that John can win if he chooses that target: a_i, if there are no neighbor targets chosen, b_i if one neighbor, and c_i if two neighbors. Could you help John choose the targets in order to maximize the number of points he can win?

Input

The program input starts with the number n ($1 \le n < 10^6$) of targets. Follows the three integers — values of a_i, b_i, and c_i for each target i ($0 \le a_i, b_i, c_i \le 1000$).

Output

Print one integer — the maximum number of points John can win.

Examples

g.in	standard output
1 3 0 0	3
1 1 2 3	1

Problem H. Red John Game

Input file:	`h.in`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

Red John has a chess table of infinite dimensions, and $n \times n$ pawns, arranged in an $n \times n$ square. The pawns can be moved horizontally or vertically, by jumping over an (horizontally or vertically) adjacent pawn, and onto the next position, only if this position is unoccupied by another pawn. Also, when a valid move occurs, the jumped (not jumping) pawn is removed. Can you help Red John figure out if there is a sequence of moves which leaves only one pawn on the table ?

Below, such a sequence of moves is illustrated, for $n = 2$. Pawns are depicted by the letter '`P`', empty spaces — by dots.

```
....    .P..    .PP.    ...P
.PP.    ..P.    ....    ....
.PP.    ..P.    ....    ....
....    ....    ....    ....
```

Input

The input contains one integer — value for n, with $0 < n < 10^9$.

Output

The output consists of 1 if there is a sequence of moves leaving only one pawn on the table, and 0 otherwise.

Examples

h.in	standard output
3	0
4	1

Problem I. Influence

Input file:	`i.in`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

In any society there are social influence relations between individuals, where a person x can influence another person y. The same is true for Softopia, a special society where social influence relations are also transitive, meaning that if x can influence y and y can influence z, then x also influences z. Moreover, the rules of social influence guarantee that if x influences any other person y, then x cannot be influenced by y as well. Using these simple rules, if a person x from Softopia wants something, then all the nodes influenced by x also want the same thing.

Although, Softopia is almost a perfect society, there is a set of certain individuals, X, that would spread false demands through the social influence mechanism used by the society. And there is also an evil entity that would want to find out which of these people should be selected to spread a demand of their own. Because the society can only select one person from X, it would like to select one that is able to influence as many people as possible from Softopia. If there is more than a person that satisfies this request, you should pick the one with the lowest identifier.

Input

The input file starts with a line containing two integers separated by one space: n ($1 \leq n \leq 5000$) the number of individuals in the society, and k, the number of elements in set X. The next line contains the elements from X, thus $k > 0$ pairwise distinct integers from 1 to n separated by one space. Then follow n lines and each line i, $1 \leq i \leq n$, contains first the identifier of the current person followed by the identifiers of the persons that can be directly influenced by person i, all of them separated by a space. The persons are labeled from 1 to n. The total number of influences in a society is less than $2.5 \cdot 10^5$. Additional whitespaces in the input file should be skipped.

Output

Print one integer — identifier of the person that satisfies the conditions mentioned above.

Examples

`i.in`	`standard output`

`5 2`
`1 2`
`1 3 4`
`2 3 4`
`3 5`
`4 5`
`5`
	`1`

`6 3`
`1 2 3`
`1 2`
`2 5`
`3 4 2`
`4 6`
`5`
`6`
	`3`

Problem J. An Idea of Mr. A

Input file:	`j.in`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 mebibytes

Mr. A proposes to his son the following problem:

"Consider two integers n_1 and n_2 such that $1 \leq n_1 < n_2 \leq 10^4$. Using the function $p : N* \rightarrow N*$, $p_n = 2^n$ for all $n \in N*$ (where $N*$ is the set of positive integers) we define the set

$$S(n_1, n_2) = \{p(p(n)) + 1 | n_1 \leq n \leq n_2\}$$

We also define a set of pairs as follows:

$$T(n_1, n_2) = \{(m_1, m_2 | m_1, m_2 \in S(n1_, n_2), m_1 < m_2\}$$

Consider the formula:

$$R(n_1, n_2) = \sum_{(m_1, m_2) \in T(n_1, n_2)} gcd(m_1, m_2)$$

where $gcd(m_1, m_2)$ is the greatest common divisor of m_1 and m_2. The problem asks to find the number $R(n_1, n_2)$."

Solve the problem proposed by Mr. A.

Input

The input file consists of a single line having the values for n_1 and n_2, separated by exactly one space.

Output

Print calculated value of $R(n_1, n_2)$.

Examples

j.in	standard output
1 34	561
15 147	8778
125 1000	383250

Problem K. Banking (Division 2 Only!)

Input file:	`k.in`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Feedback:	

The bankers of Somebank have collected day-to-day data of the daily profits (and losses) of the shares they hold. Based on these numbers they decide to calculate which days would have been the most profitable to buy and sell their shares, so they can compare that with their actual gain.

Your task is to write a program that computes the optimal «run» in the sequence of numbers, the subsequence that maximizes the profit. The subsequence you compute is represented by the 1-based indexes of the first and last numbers in the subsequence. We are asking for exactly one date to buy and one date to sell shares since otherwise the solution would be simple: keep your shares on dates that the profit is non-negative.

Input

Input file has the following format:

- One line with a single integer N, $(1 \le N \le 10^6)$ — the length of the sequence to follow.
- One line with N integers p_i $(-1000 \le p_i \le 1000)$ — the profit (or loss) on date i.

The integers are separated by single spaces. At least one of the integers is positive.

Output

Output file contains a single line with two integers k and l $(1 \le k \le l \le N)$, such that the sum of the k-th until the k-th integer is maximized, boundaries included. When k and l can be selected by different ways, choose minimal k and minimal l.

Example

k.in	standard output
11 -3 1 -1 2 3 1 -1 2 -3 -5 7	2 8
9 1 -2 3 -1 -1 3 -2 2 -4	3 6

Problem L. Dimensions (Division 2 Only!)

Input file:	`l.in`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Feedback:	

Teacher has given his pupils a table of widths, heights, lengths and volumes of some cuboid. Each row on the table has one of the 4 values missing; the pupils — and your program — have to work out the missing value and write it in the table such that the values on each line represent the width, height, length and volume of one cuboid.

Input

Input is a series of lines, each containing 4 integers: w, h, l and v representing the width, height, length and volume of a cuboid in that order. The integers are separated by a single space. $0 < l, w, h < 100, 0 < v < 10^5$. In each row, one of the values has been replaced by a zero. The final row contains 0 0 0 0 and should not be processed.

Output

Output is the same series of lines but with the zero in each line replaced by the correct value for length, width, height or volume as appropriate. It is guaranteed that the new value is always an integer.

Example

`l.in`	`standard output`

`1 0 2 6`
`5 5 5 0`
`0 2 2 80`
`8 0 9 576`
`0 0 0 0`
	`1 3 2 6`
	`5 5 5 125`
	`20 2 2 80`
	`8 9 9 576`

Problem M. Greatest Product (Division 2 Only!)

Input file:	`m.in`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 Mebibytes
Feedback:	

Game Master announces positive integer N. Player needs to calculate product of digits for each positive integer up to N and report the greatest product. All answers should be known beforehand to allow interactive communication during a game. This causes problems since N could be relatively big ($1 \leq N \leq 2 \cdot 10^9$). Game Master is not good using computers; he asks you to write a program which having positive integer N will find correct answer.

Input

Input file contains one integer N

Output

Output file should contain greatest product for given N.

Examples

m.in	standard output
1	1
101090000	43046721
28994	10368
4876	2268
2789	1008