Problem A. Maximum Subarray Sum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Chino has an array a of length N. He wants to perform at most M swapping operations: choose a pair of elements and swap their values (not necessarily adjacent). Then he chooses $\left\lfloor \frac{N}{K} \right\rfloor$ non-overlapping subarrays of length at least K.

What is the maximum sum of elements for all selected subarrays?

A subarray is a contiguous part of the array. An array that is inside another array.

Input

The first line contains three positive integers $N, K, M (1 \le K \le N \le 10^4, 0 \le M \le 20)$.

The next line contains N positive integers $a_i (-10^5 \le a_i \le 10^5)$.

Output

A single line containing one integer representing the answer.

Examples

standard input	standard output
11 3 0 -1 1 1 -1 1 1 1 1 -1 1 1	6
11 3 1 -1 1 1 -1 1 1 1 1 -1 1 1	7
11 3 5 9 8 7 6 5 8 4 6 5 4 8	70

Problem B. Lottery

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Chino has purchased N lottery tickets, with each ticket priced at $\frac{M}{K}$ Yuan, **where $\frac{M}{K}$ is a real number not exceeding 2.** The aggregate expense amounts to $\frac{N \cdot M}{K}$ Yuan. The first ticket can win either 0 or 1 Yuan with equal probability, the second ticket can win 0, 1, or 2 Yuan with equal probability, and so forth. The i-th ticket can win any from the set $\{0, 1, 2, ..., i\}$ Yuan, each with an equal probability.

Chino is now interested in determining the probability that the total winnings from the N tickets will surpass the total cost of purchasing them.

Input

The first line contains a positive integer $T(1 \leq T \leq 10^5)$ — the number of test cases.

For each test case:

input three positive integers $N, M, K(1 \leq N \leq 10^5, 1 \leq K \leq 10^5, 1 \leq M \leq min(10^5, 2 \cdot K))$ — there are N lottery tickets, and the price of each ticket is $\frac{M}{K}$.

Output

For each test case, output the probability as an integer in the form of $P \cdot Q^{-1} \pmod{10^9 + 7}$, where Q^{-1} is the modular multiplicative inverse of Q with respect to the modulus $10^9 + 7$.

Examples

standard input	standard output
1 2 1 2	500000004
6 100000 1 1 100000 1 2 99999 1 1 99998 1 1 2 3 2 114 114 541	883883568 457694755 925667272 222808310 0 852920407
3 511 61754 38055 682 11918 11494 776 63852 33072	211077063 959161364 92506168

Problem C. Array Sorting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Chino has an integer array a of length N, with indices from 0 to $N-1$, and she wants to sort it in non-decreasing order.

Specifically, she can perform a magical spell:

Choose $2 \cdot M$ **distinct** indices, paired as $(id_0, id_1), (id_2, id_3), ..., (id_{2 \cdot M-2}, id_{2 \cdot M-1})$, for all $i \in [0, M)$; if $a_{id_{2 \cdot i}} > a_{id_{2 \cdot i+1}}$, then swap the values of $a_{id_{2 \cdot i}}$ and $a_{id_{2 \cdot i+1}}$.

Now, you need to construct a feasible plan so that regardless of the elements in the array, they can be sorted into non-decreasing order.

You are required to perform the magical spell no more than 200 times.

The jury will test your algorithm several times. If your algorithm passes all tests, the problem is considered solved; otherwise, the answer will be deemed incorrect.

The test cases include:

- Sorted array in descending order;
- Sorted array with k random swap operations using the same random seed($k = 1, 5, 10, 50, 100, 1000, 10000$);
- Sorted array with cyclic shift;
- All permutations with the array size less than 10;
- A completely random array with the same random seed in each judgment.

Input

Only one line with a positive integer $N(1 \le N \le 10^4)$.

Output

The first line should output an integer, representing the number of times the magical spell is cast, *ans*.

If *ans* is not 0, then continue to output *ans* lines, each line starting with a positive integer $M(1 \le M \le \frac{N}{2})$— the number of index pairs chosen for the magical operation.

Then continue to output $2 \cdot M$ integers representing the chosen indices.

Examples

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

Problem D. Interval Selection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

YangQiShaoNian has an array of length n, a subarray $[l, r]$ in the array is good if and only if each element in $a_l, a_{l+1}, \ldots a_r$ appears exactly k times in the current interval.

For example, for $a = [1, 1, 2, 3, 2, 3, 1]$ and $k = 2$, the intervals $[1, 2]$, $[3, 6]$, $[1, 6]$, etc., are all good. However, $[1, 3]$ does not meet the condition because the element 2 appears only once, and $[1, 7]$ does not meet the condition because the element 1 appears 3 times.

Please help YangQiShaoNian to find the number of good intervals that can be selected.

Input

The first line contains an integer $T(1 \le T \le 10^5)$, indicating the number of test cases.

For each test case:

The first line contains two integers $n(1 \le n \le 2 \cdot 10^5), k(1 \le k \le n)$.

The second line contains n integers $a_1 \ldots a_n(0 \le a_i \le 10^9)$ — each element of the array.

It is guaranteed that the total sum of n for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case:

Only output one line containing an integer, representing the answer.

Example

standard input	standard output
4 3 2 1 2 2 3 1 1 2 3 6 2 1 1 4 5 1 4 6 3 1 2 1 1 1 1	1 6 1 2

Problem E. Flowers

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	512 megabytes

Emofunc has a tree with n nodes, and each node has a color. A flower is a connected subgraph of the tree in which all the nodes of degree 1 have the same color. If a connected subgraph is a flower, its petal count is defined as the number of nodes of degree 1 within that subgraph. Note that a connected subgraph consisting of only one node is also considered a flower, with a petal count of 0.

Now, Emofunc provides you with the color of each node and a positive integer k ($1 \leq k \leq n$). He wants to know how many flowers in the tree have a petal count no more than k. Output the answer modulo 998244353.

Input

The 1-st line contains two integers n ($1 \leq n \leq 10^5$) and k ($1 \leq k \leq n$) – the number of nodes and the upper bound of petal count.

The 2-nd line contains n integers $c_1, c_2, ..., c_n$ ($1 \leq c_i \leq n$), where c_i is the color of node i.

Then $n-1$ lines follow. Each line contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$) – an edge between node u_i and v_i. It is guaranteed that these edges form a tree.

Output

An integer – the number of flowers whose petal count $\leq k$, modulo 998244353.

Examples

standard input
5 3 3 1 3 2 3 1 5 4 1 2 4 3 4
standard output
8

standard input
20 4 2 2 1 1 1 1 2 1 3 1 3 3 3 2 2 3 2 3 3 1 17 7 1 7 20 19 1 4 13 18 3 20 12 2 19 16 18 3 5 6 12 15 10 5 18 11 8 9 19 7 12 7 6 13 4 9 1 14
standard output
108

Problem F. Teleportation

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

In the year 2124, humans have established $10^{100} - 2$ space stations in space, numbered $3, 4, \cdots, 10^{100}$. They invent a technology that allows teleportation between space stations. For two space stations with numbers x and y, if there exists a positive integer z such that $\frac{1}{x} + \frac{1}{y} = \frac{1}{z}$, it is possible to teleport from space station x to space station y.

Now, emofunc wants to explore the space stations with numbers $\leq 10^9$. Each time, he will move from space station x to space station y only through teleportation. Since each teleportation consumes a significant amount of energy, he hopes to limit the number of teleportations to no more than 150 times. Please provide a plan for each query.

Input

The first line contains an integer T ($1 \leq T \leq 50$) — the number of queries. Then T lines follow, each line contains two integers x, y ($3 \leq x, y \leq 10^9, x \neq y$) — the number of starting and end station.

Output

For each test case, first print a line containing an integer m — the number of teleportations.

Then, print another line containing $m + 1$ integers $v_0, v_1, ..., v_m$ with adjacent integers separated by a space.

They should satisfy that $m \leq 150$, $v_0 = x$, $v_m = y$, $3 \leq v_i \leq 10^{100}$ for $0 \leq i \leq m$ and for all $0 \leq i < m$, there exists a positive integer z_i such that $\frac{1}{v_i} + \frac{1}{v_{i+1}} = \frac{1}{z_i}$.

Example

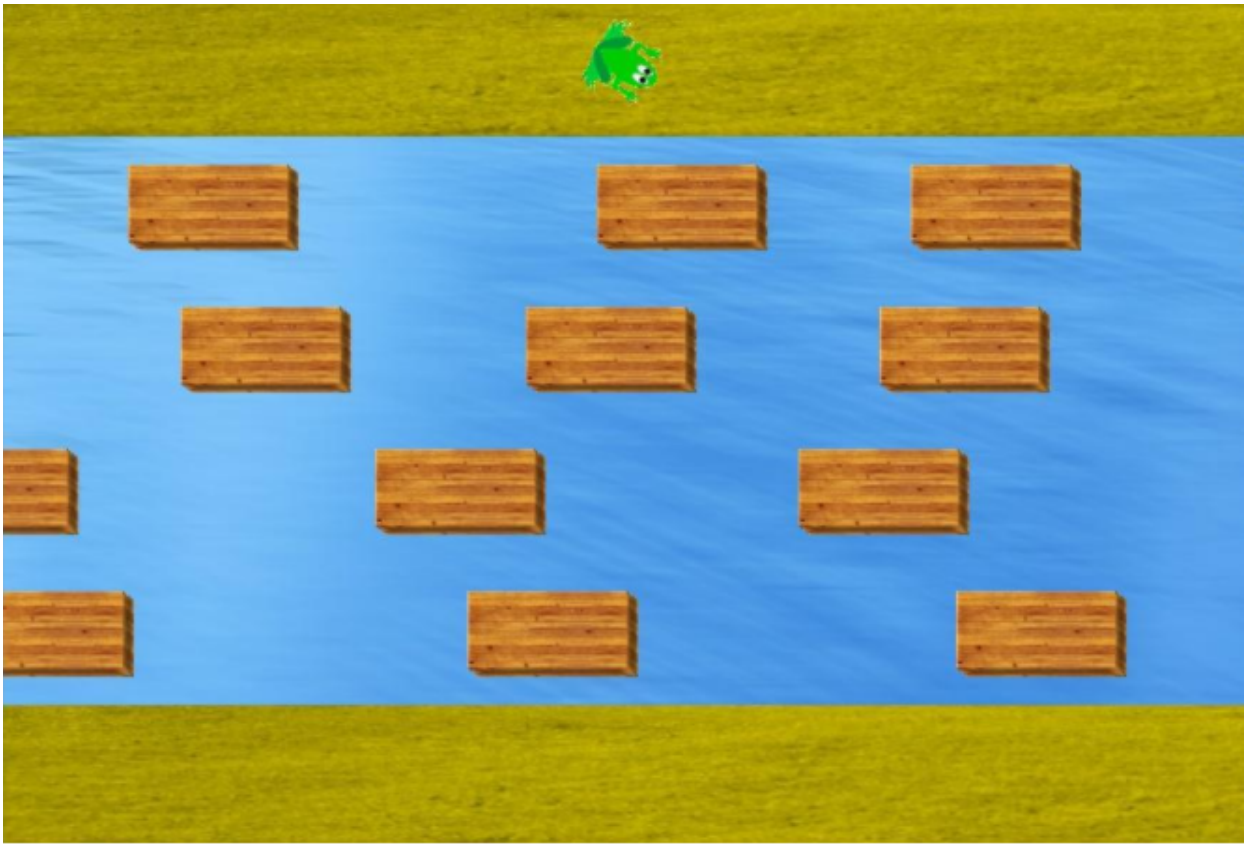
standard input
3 3 4 3 5 114514 1919810
standard output
3 3 6 12 4 4 3 6 30 20 5 5 114514 6985354 166408 4992240 5724240 1919810

Problem G. Frog Crossing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

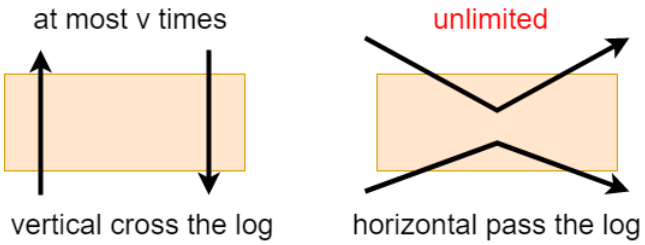
Chino is playing a game called "Frog Crossing".

There is a group of frogs that need to cross a river with a distance of N between the two banks. They plan to use M planks floating on the river.



The river can be considered as a 2D plane. The frogs start on the coordinate axis at $x = 0$ and want to move to the opposite bank at $x = N + 1$.

Specifically, the i-th plank is a rectangle of size $1 \times l_i$ with its left endpoint at coordinates (x_i, y_i).



To prevent too many frogs from crossing vertically and causing the plank to flip, it is stipulated that at most v_i frogs can **vertical cross** the i-th plank, but an unlimited number of frogs can **horizontal pass** it.

Vertical cross is defined as: a frog enters from below a plank and exits from above; or a frog enters from above a plank and exits from below.

Horizontal pass is defined as: a frog enters from below a plank and exits from below; or a frog enters

from above a plank and exits from above.

For any two planks, it is guaranteed that they do not intersect or touch each other at the same x coordinate.

Frogs can move between planks i and j iff $|x_i - x_j| = 1$ and $[y_i, y_i + l_i - 1] \cap [y_j, y_j + l_j - 1] \neq \emptyset$.

Now there are M queries. For the i-th query, assume that the i-th plank does not allow frogs to cross vertically (i.e., v_i is set to 0 in the i-th query). Determine the number of frogs that can eventually reach the opposite bank for each query.

Input

The first line contains two positive integers N and M ($1 \leq N \leq 10^5, 1 \leq M \leq 2 \times 10^5$) — the distance between the two banks and the number of planks, respectively.

The next M lines each contain four integers x_i, y_i, l_i, v_i ($1 \leq x_i \leq N, 1 \leq y_i, l_i, v_i \leq 10^9$).

Output

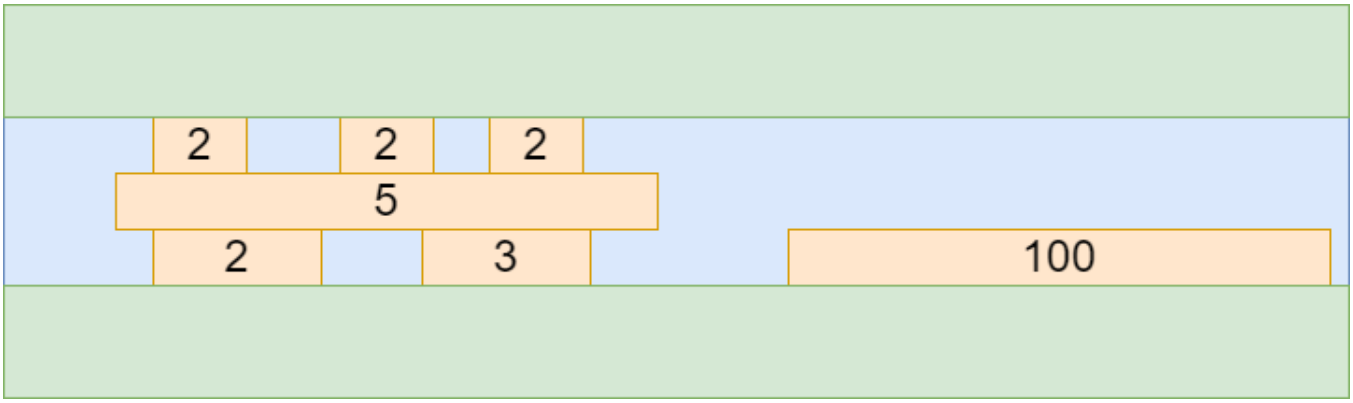
Output M lines, each representing the answer to the i-th query for $i = 1, 2, 3, ..., M$.

Examples

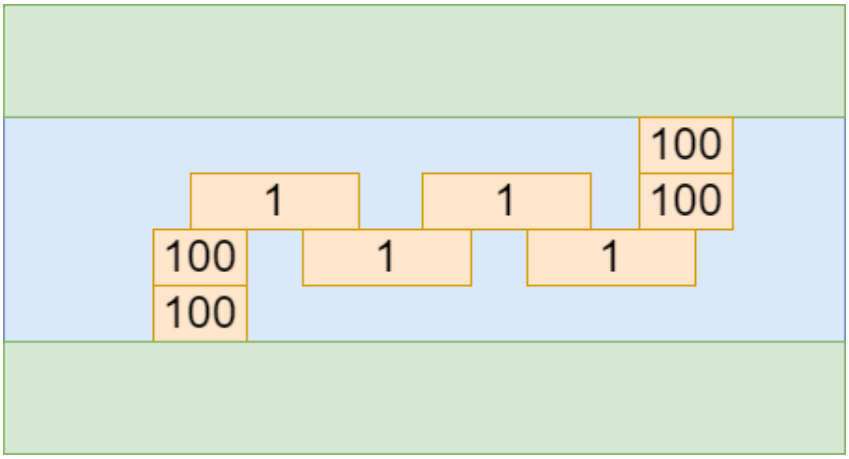
standard input	standard output
3 7 1 100 100 100 3 5 1 2 1 1 1 2 3 1 1 2 3 3 1 2 1 6 1 3 2 1 10 5	5 4 3 4 4 2 0
4 8 1 1 2 100 2 1 2 100 3 2 3 1 2 4 3 1 3 6 3 1 2 8 3 1 3 10 2 100 4 10 2 100	0 0 100 100 100 100 0 0

Note

For example 1.



For example 2, note that when a frog horizontal passes the plank, it can be unlimited.



Problem H. Database

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

YangQiShaoNian is a fan of data structures. After studying the course CMU15 – 445 Introduction to Database Systems, he decided to lead you in front of the screen to create a simple SQL statement executor.

This database can be abstracted as a table with a variable number of rows and exactly n columns.

Each column of the table has a unique field name, and the field names are all different.

You will need to handle various types of SQL statements. The statement formats are as follows:

1. SQL Statements

1.1. `insert` Statement

Syntax

`insert`(field value1,field value2, . . . ,field value$n$)

Explanation

Inserts a new record into the table, which has exactly n parameters. The i-th parameter represents the value of the i-th field, and each parameter is a string. The string contains only lowercase and uppercase letters, numbers, and underscores.

Example

If the table is currently:

Field	`id`	`name`	`birth_place`
Data	`a`	`Alice`	`Beijing`
	`b`	`Bob`	`Beijing`

After executing `insert`(`c`, `Yangqishaonian`, `Nowcoder`), the table becomes:

Field	`id`	`name`	`birth_place`
Data	`a`	`Alice`	`Beijing`
	`b`	`Bob`	`Beijing`
	`c`	`Yangqishaonian`	`Nowcoder`

1.2. `select` Statement

Syntax

`select`(output field name,condition field name,condition value)

Explanation

The output field name and condition field name are among the given n field names.

The condition value is a string. The string contains only lowercase and uppercase letters, numbers, and underscores.

For a certain data, if its condition field is equal to the condition value, then add the corresponding data item of this data to the returned set.

Return Value

The result of the query is a sorted set of strings, sorted in the order of insertion.

Example

If the table is currently:

Field	id	name	birth_place
Data	a	Alice	Beijing
	b	Bob	Beijing
	c	Yangqishaonian	Nowcoder

At this time, the return value of `select(name,birth_place,Beijing)` is a set of 2 strings, where the first string is `Alice` and the second string is `Bob`.

1.3. `delete` Statement

Syntax

`delete`(condition field name, condition value)

Explanation

The condition field name is among the given n field names.

The condition value is a string. The string contains only lowercase and uppercase letters, numbers, and underscores.

For a certain data, if its condition field is equal to the condition value, then it is deleted.

Example

If the table is currently:

Field	id	name	birth_place
Data	a	Alice	Beijing
	b	Bob	Beijing
	c	Yangqishaonian	Nowcoder

After executing `delete(birth_place,Beijing)`, the table becomes:

Field	id	name	birth_place
Data	c	Yangqishaonian	Nowcoder

1.4. `select_in` Statement

Syntax

`select_in`(output field name,condition field name,condition value set)

Explanation

The output field name and condition field name are among the given n field names.

The condition value set is a set of strings.

For a certain data, if its condition field appears in the condition value set, then add the corresponding data item of this data to the returned set.

Return Value

The result of the query is a sorted set of strings, sorted in the order of insertion.

Note

For the third parameter, it can only nest `select` or `select_in` statements.

Example

Field	A	B	C	D
Data	4	2	3	x
	3	1	2	y
	1	4	2	z

The return value of `select_in`(`D`, `A`, `select`(`B`, `C`, 2)) is a set of 2 strings, where the first string is `x` and the second string is `z`.

1.5. `delete_in` Statement

Syntax

`delete_in`(condition field name,condition value set)

Explanation

The condition field name is among the given n field names.

The condition value set is a set of strings.

For a certain data, if its condition field appears in the condition value set, then it is deleted.

Note

For the second parameter, it can only nest `select` or `select_in` statements.

Example

Field	A	B	C	D
Data	4	2	3	x
	3	1	2	y
	1	4	2	z

After executing `delete_in`(`A`, `select`(`B`, `C`, 2)), the table becomes:

Field	A	B	C	D
Data	3	1	2	y

2. Transaction Statements

In the database, each SQL statement must **belong to exactly one transaction**.

The format of each transaction is as follows:

1) `begin`() Statement — Indicates the start of a transaction

2) Several SQL statements... — All these SQL statements belong to the current transaction

3) `commit`() Statement or `abort`() Statement — Indicates the end of a transaction

The next three subsections will introduce the three types of statements mentioned above.

2.1. `begin` Statement

Syntax

`begin`()

Explanation

Indicates the start of a new database transaction, and the first statement must be `begin`().

Next, several (or 0) SQL statements will be given.

All SQL statements before the next `commit`() or `abort`() statement belong to this transaction.

It is guaranteed that the next **transaction statement** must be `commit`() or `abort`().

2.2. `commit` Statement

Syntax

`commit`()

Explanation

All changes to the database caused by the SQL statements between the previous `begin`() statement and this statement need to be preserved.

If this statement is not the last statement, it is guaranteed that the next statement must be `begin`().

2.3. `abort` Statement

Syntax

`abort`()

Explanation

All changes to the database caused by the SQL statements between the previous `begin`() statement and this statement need to be discarded.

If this statement is not the last statement, it is guaranteed that the next statement must be `begin`().

Input

The first line contains two integers $n(1 \leq n \leq 1000), q(3 \leq q \leq 3000)$, representing the number of field names and the number of statements, respectively.

The second line contains n strings $key_1, \ldots, key_n$, representing the field names. Each field name is guaranteed to contain only lowercase and uppercase letters, numbers, and underscores, and the total length of all field names does not exceed 5000, there are no duplicate field names.

The next q lines each contain a string representing the statement to be executed. Each statement satisfies the following conditions:

- The length does not exceed 2000;
- The format strictly complies with the syntax constraints mentioned above;
- The number of left parentheses '(' does not exceed 10.

Output

For each statement starting with `select` or `select_in`:

The first line outputs an integer x, representing the number of entries in the returned set.

Next:

- If $x = 0$, then you do not need to output anything for this statement;
- If $x \geq 1$, to reduce the amount of output, you only need to output three lines: the first line is the first result, the second line is the $\lceil \frac{x}{2} \rceil$-th result, and the third line is the x-th result.

For each statement starting with `delete` or `delete_in`:

Only output one line containing an integer, representing the number of entries deleted.

Examples

standard input
3 13 id name birth_place begin() insert(a,Alice,Beijing) insert(b,Bob,Beijing) insert(c,Yangqishaonian,Nowcoder) select(name,birth_place,Beijing) commit() begin() delete(birth_place,Beijing) select(name,birth_place,Beijing) abort() begin() select(name,birth_place,Beijing) abort()
standard output
2 Alice Alice Bob 2 0 2 Alice Alice Bob

standard input

```
5 11
a b c d e
begin()
insert(1,2,3,5,5)
insert(4,1,2,3,5)
insert(4,4,1,2,3)
commit()
begin()
select(b,e,5)
select_in(a,c,select(b,e,5))
select_in(d,a,select_in(a,c,select(b,e,5)))
delete_in(c,select_in(d,a,select_in(a,c,select(b,e,5))))
commit()
```

standard output

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

Problem I. Fight Against the Monster

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

On a mysterious planet, YangQiShaoNian and Keith must fight a terrifying monster.

Now, several war machines need to be built to combat the monster. The monster starts with a health of h, and it will die when its health falls to 0 or below.

Each machine has two functions—"Fight" and "Create"—described as follows:

Fight: For a given machine, using the"Fight" function will reduce the monster's health by 1, but the machine will be permanently damaged and lose all functionality;

Create: To use the"Create" function, first, m machines must be chosen, and this function must be used on them simultaneously, which will then produce k new machines. Note that each machine can only use the "Create" function once, but can still use the "Fight" function afterwards.

Please calculate the minimum number of initial war machines required to defeat the monster.

Input

The first line contains an integer $T(1 \le T \le 2 \cdot 10^5)$, indicating the number of test cases. For each test case:

Only one line is input, containing three integers $m, k, h(1 \le k \le m \le 10^6, 0 \le h \le 10^9)$, representing the number of machines required to use the "Create" function, the number of machines produced by using the "Create" function, and the monster's initial health.

Output

For each test case:

Output only one line containing an integer, representing the answer.

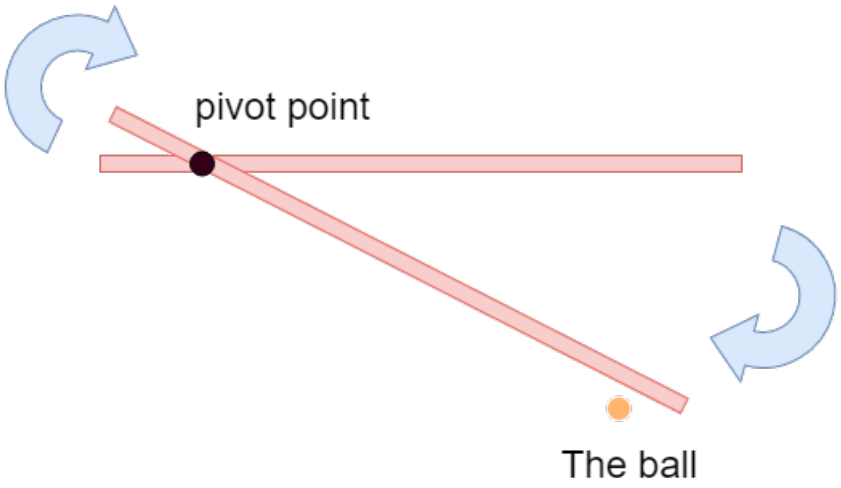
Example

standard input	standard output
2 2 1 7 97 33 114514	4 75574

Problem J. Ball

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Chino is considering such a geometric problem.



In the two-dimensional Cartesian coordinate system, there is a long stick with negligible width and a length of l perpendicular to the y-axis, with the left endpoint at the origin $O(0,0)$.

A small ball, which can be seen as a point, is located at $P(x,y)$.

Chino wants to know whether there is a pivot point on the long stick, so that during the rotation of the stick around the pivot point, it can hit the small ball. Please help Chino find any pivot point that satisfies the condition, or tell her that such a pivot point does not exist.

Iff the stick or its endpoint passes through point $P(x,y)$ during rotation, the stick can hit the ball.

Input

The first line contains a positive integer $T(1 \le T \le 10^4)$ — the number of test cases.

For each test case:

Input a line with three integers $l, x, y(1 \le l \le 10^5, -10^5 \le x, y \le 10^5)$ — the length of the long stick and the coordinates of the small ball.

Output

If there is a pivot point that satisfies the condition, first print "Yes", followed by the x-coordinate of the pivot point on the next line, you can give any real number in the range $[0, l]$; otherwise, print "No".

Examples

standard input	standard output
3 5 3 1 5 2 1 5 3 6	Yes 3.114514 Yes 2.114514 No
2 17 -8 15 17 17 18	Yes 0 No

Problem K. Strings, Subsequences, Reversed Subsequences, Prefixes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

For a string $S = S_0S_1...S_{n-1}$ of length n, its reversed string is defined as $R(S) = S_{n-1}S_{n-2}...S_0$.

Chino has two strings S and T, she wants to know how many **distinct** non-empty subsequences seq there are in S such that T is simultaneously a prefix of both seq and $R(seq)$.

Since the answer may be large, please output the result modulo $10^9 + 7$.

Some definitions are seen in the notes for explanation.

Input

The first line contains two positive integers N and M ($1 \leq N, M \leq 10^6$) — the lengths of S and T.

Then the next two lines contain the contents of strings S and T ($S_i, T_i \in [a, z]$).

Output

Only one line containing a non-negative integer — the answer to the problem modulo $10^9 + 7$.

Example

standard input	standard output
7 2 abababa ab	8

Note

seq can be “aba”,“abba”,“ababa”,“abbba”,“abaaba”,“ababba”,“abbaba”,“abababa”.

A non-empty subsequence of string S is defined as a new string obtained by deleting one or more characters from the string (while keeping at least one character).

The prefix of string S is defined as the part left after deleting all characters from a certain position to the end of the string.

Two strings are **distinct** iff their lengths or contents are different.