Codeforces β
Sponsored by Telegram

Divide by Zero 2018 and Codeforces Round #474 (Div. 1 + Div. 2, combined)

A. Check the string

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A has a string consisting of some number of lowercase English letters 'a'. He gives it to his friend B who appends some number of letters 'b' to the end of this string. Since both A and B like the characters 'a' and 'b', they have made sure that at this point, **at least one 'a' and one 'b'** exist in the string.

B now gives this string to C and he appends some number of letters 'c' to the end of the string. However, since C is a good friend of A and B, the number of letters 'c' he appends is equal to the number of 'a' or to the number of 'b' in the string. It is also possible that the number of letters 'c' equals both to the number of letters 'a' and to the number of letters 'b' at the same time.

You have a string in your hands, and you want to check if it is possible to obtain the string in this way or not. If it is possible to obtain the string, print "YES", otherwise print "NO" (without the quotes).

Input

The first and only line consists of a string $$$S$$$ ($$$ 1 \le |S| \le 5\,000 $$$). It is guaranteed that the string will only consist of the lowercase English letters 'a', 'b', 'c'.

Output

Print "YES" or "NO", according to the condition.

Examples

input
```
aaabccc
```
output
```
YES
```

input
```
bbacc
```
output
```
NO
```

input
```
aabc
```
output
```
YES
```

Note

Consider first example: the number of 'c' is equal to the number of 'a'.

Consider second example: although the number of 'c' is equal to the number of the 'b', the order is not correct.

Consider third example: the number of 'c' is equal to the number of 'b'.

B. Minimize the error

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two arrays A and B, each of size n. The error, E, between these two arrays is defined $E = \sum_{i=1}^{n}(a_i - b_i)^2$. You have to perform **exactly** k_1 operations on array A and **exactly** k_2 operations on array B. In one operation, you have to choose one element of the array and increase or decrease it by 1.

Output the minimum possible value of error after k_1 operations on array A and k_2 operations on array B have been performed.

Input

The first line contains three space-separated integers n ($1 \le n \le 10^3$), k_1 and k_2 ($0 \le k_1 + k_2 \le 10^3$, k_1 and k_2 are non-negative) — size of arrays and number of operations to perform on A and B respectively.

Second line contains n space separated integers $a_1, a_2, \ldots, a_n$ ($-10^6 \le a_i \le 10^6$) — array A.

Third line contains n space separated integers $b_1, b_2, \ldots, b_n$ ($-10^6 \le b_i \le 10^6$)— array B.

Output

Output a single integer — the minimum possible value of $\sum_{i=1}^{n}(a_i - b_i)^2$ after doing exactly k_1 operations on array A and exactly k_2 operations on array B.

Examples

input

```
2 0 0
1 2
2 3
```

output

```
2
```

input

```
2 1 0
1 2
2 2
```

output

```
0
```

input

```
2 5 7
3 4
14 4
```

output

```
1
```

Note

In the first sample case, we cannot perform any operations on A or B. Therefore the minimum possible error $E = (1 - 2)^2 + (2 - 3)^2 = 2$.

In the second sample case, we are required to perform exactly one operation on A. In order to minimize error, we increment the first element of A by 1. Now, $A = [2, 2]$. The error is now $E = (2 - 2)^2 + (2 - 2)^2 = 0$. This is the minimum possible error obtainable.

In the third sample case, we can increase the first element of A to 8, using the all of the 5 moves available to us. Also, the first element of B can be reduced to 8 using the 6 of the 7 available moves. Now $A = [8, 4]$ and $B = [8, 4]$. The error is now $E = (8 - 8)^2 + (4 - 4)^2 = 0$, but we are still left with 1 move for array B. Increasing the second element of B to 5 using the left move, we get $B = [8, 5]$ and $E = (8 - 8)^2 + (4 - 5)^2 = 1$.

C. Subsequence Counting

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Pikachu had an array with him. He wrote down all the non-empty subsequences of the array on paper. Note that an array of size n has $2^n - 1$ non-empty subsequences in it.

Pikachu being mischievous as he always is, removed all the subsequences in which `Maximum_element_of_the_subsequence - Minimum_element_of_subsequence` $\ge d$

Pikachu was finally left with X subsequences.

However, he lost the initial array he had, and now is in serious trouble. He still remembers the numbers X and d. He now wants you to construct any such array which will satisfy the above conditions. All the numbers in the final array should be positive integers less than 10^{18}.

Note the number of elements in the output array should not be more than 10^4. If no answer is possible, print -1.

Input

The only line of input consists of two space separated integers X and d ($1 \le X, d \le 10^9$).

Output

Output should consist of two lines.

First line should contain a single integer n ($1 \le n \le 10\,000$)— the number of integers in the final array.

Second line should consist of n space separated integers — $a_1, a_2, \dots, a_n$ ($1 \le a_i < 10^{18}$).

If there is no answer, print a single integer `-1`. If there are multiple answers, print any of them.

Examples

input
```
10 5
```
output
```
6
5 50 7 15 6 100
```

input
```
4 2
```
output
```
4
10 100 1000 10000
```

Note

In the output of the first example case, the remaining subsequences after removing those with `Maximum_element_of_the_subsequence - Minimum_element_of_subsequence` ≥ 5 are $[5], [5, 7], [5, 6], [5, 7, 6], [50], [7], [7, 6], [15], [6], [100]$. There are 10 of them. Hence, the array $[5, 50, 7, 15, 6, 100]$ is valid.

Similarly, in the output of the second example case, the remaining sub-sequences after removing those with `Maximum_element_of_the_subsequence - Minimum_element_of_subsequence` ≥ 2 are $[10], [100], [1000], [10000]$. There are 4 of them. Hence, the array $[10, 100, 1000, 10000]$ is valid.

D. Full Binary Tree Queries

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a full binary tree having infinite levels.

Each node has an initial value. If a node has value x, then its left child has value $2 \cdot x$ and its right child has value $2 \cdot x + 1$.

The value of the root is 1.

You need to answer Q queries.

There are 3 types of queries:

1. Cyclically shift the ***values*** of all nodes on the same level as node with value X by K units. (The values/nodes of any other level are not affected).
2. Cyclically shift the ***nodes*** on the same level as node with value X by K units. (The subtrees of these nodes will move along with them).
3. Print the value of every node encountered on the simple path from the node with value X to the root.

Positive K implies right cyclic shift and negative K implies left cyclic shift.

It is guaranteed that atleast one type 3 query is present.

Input

The first line contains a single integer Q ($1 \le Q \le 10^5$).

Then Q queries follow, one per line:

- Queries of type 1 and 2 have the following format: $T\ X\ K$ ($1 \le T \le 2$; $1 \le X \le 10^{18}$; $0 \le |K| \le 10^{18}$), where T is type of the query.
- Queries of type 3 have the following format: $3\ X$ ($1 \le X \le 10^{18}$).

Output

For each query of type 3, print the values of all nodes encountered in descending order.

Examples

input
```
5
3 12
1 2 1
3 12
2 4 -1
3 8
```
output
```
12 6 3 1
```

```
12 6 2 1
8 4 2 1
```

input

```
5
3 14
1 5 -3
3 14
1 3 1
3 14
```

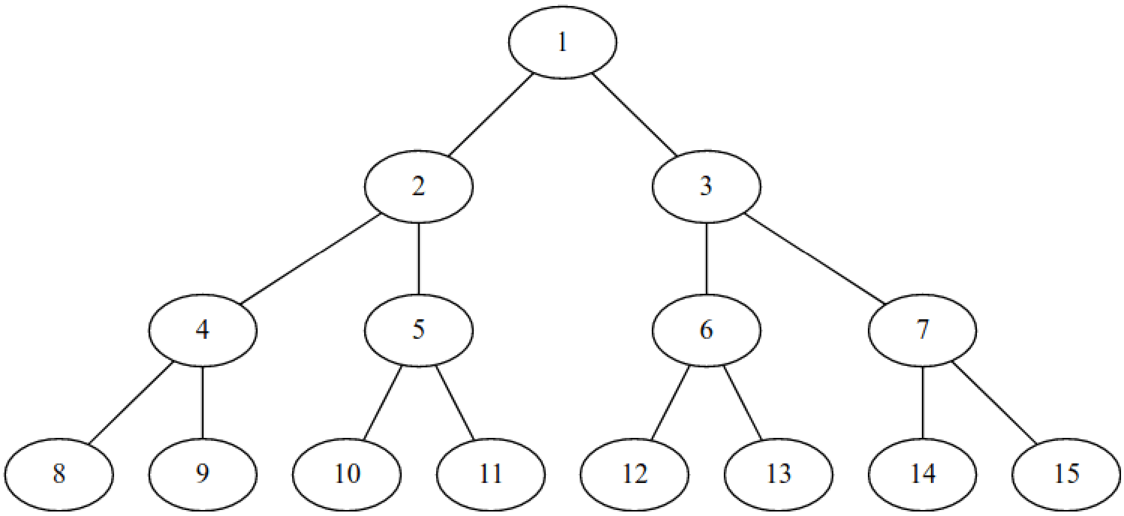
output

```
14 7 3 1
14 6 3 1
14 6 2 1
```

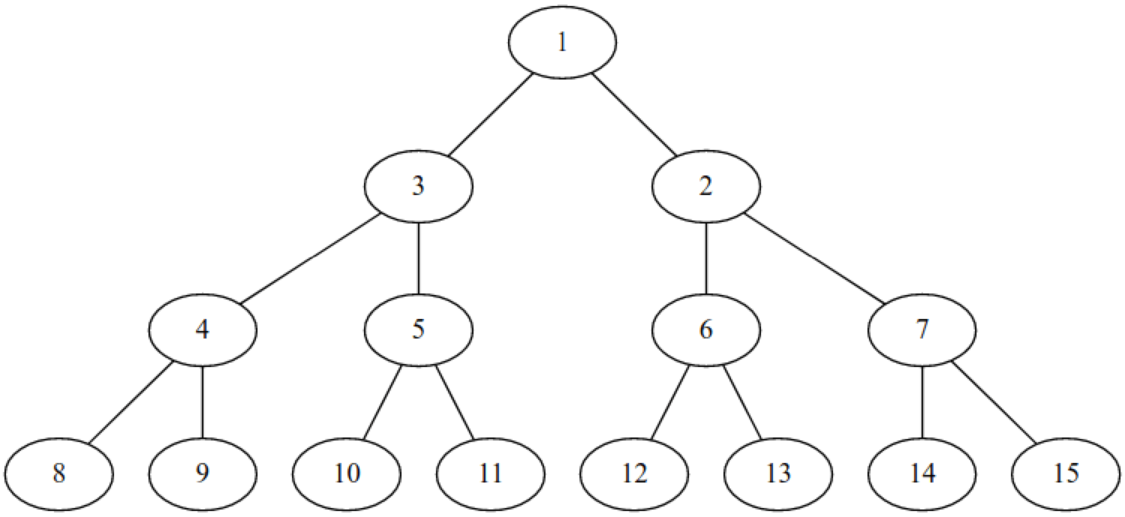
Note

Following are the images of the first 4 levels of the tree in the first test case:

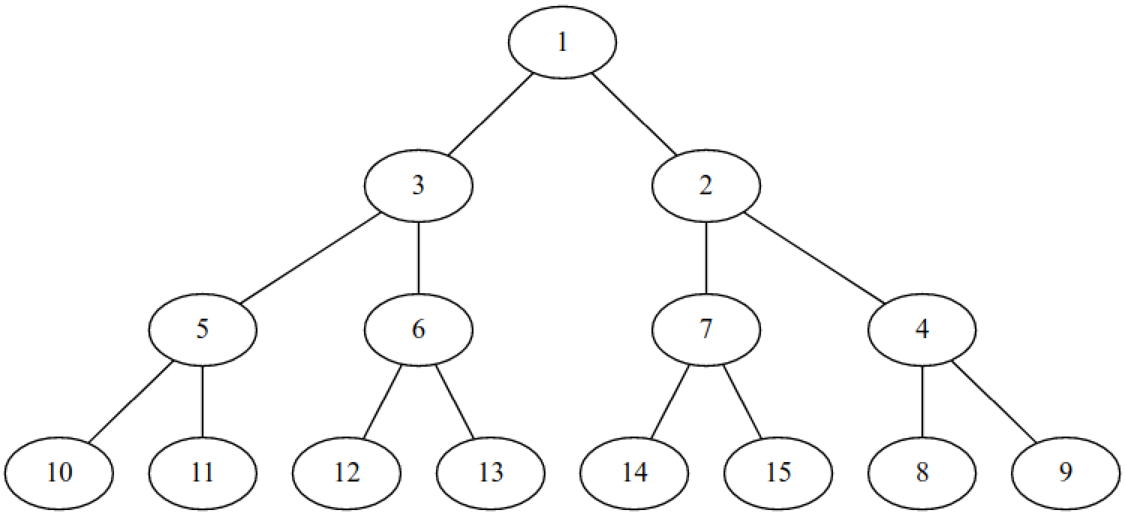
Original:



After query *1 2 1*:



After query *2 4 -1*:



E. Alternating Tree

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Given a tree with $$$n$$$ nodes numbered from $$$1$$$ to $$$n$$$. Each node $$$i$$$ has an associated value $$$V_i$$$.

If the simple path from $$$u_1$$$ to $$$u_m$$$ consists of $$$m$$$ nodes namely $$$u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow \dots u_{m-1} \rightarrow u_{m}$$$, then its alternating function $$$A(u_{1},u_{m})$$$ is defined as $$$A(u_{1},u_{m}) = \sum\limits_{i=1}^{m} (-1)^{i+1} \cdot V_{u_{i}}$$$. A path can also have $$$0$$$ edges, i.e. $$$u_{1}=u_{m}$$$.

Compute the sum of alternating functions of all unique simple paths. Note that the paths are directed: two paths are considered different if the starting vertices differ or the ending vertices differ. The answer may be large so compute it modulo $$$10^{9}+7$$$.

Input

The first line contains an integer $$$n$$$ $$$(2 \leq n \leq 2\cdot10^{5})$$$ — the number of vertices in the tree.

The second line contains $$$n$$$ space-separated integers $$$V_1, V_2, \ldots, V_n$$$ ($$$-10^9\leq V_i \leq 10^9$$$) — values of the nodes.

The next $$$n-1$$$ lines each contain two space-separated integers $$$u$$$ and $$$v$$$ $$$(1\leq u, v\leq n, u \neq v)$$$ denoting an edge between vertices $$$u$$$ and $$$v$$$. It is guaranteed that the given graph is a tree.

Output

Print the total sum of alternating functions of all unique simple paths modulo $$$10^{9}+7$$$.

Examples

input

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

output

```
40
```

input

```
8
-2 6 -4 -4 -9 -3 -7 23
8 2
2 3
1 4
6 5
7 6
4 7
5 8
```

output

```
4
```

Note

Consider the first example.

A simple path from node $$$1$$$ to node $$$2$$$: $$$1 \rightarrow 2$$$ has alternating function equal to $$$A(1,2) = 1 \cdot (-4)+(-1) \cdot 1 = -

5$$$.

A simple path from node $$$1$$$ to node $$$3$$$: $$$1 \rightarrow 3$$$ has alternating function equal to $$$A(1,3) = 1 \cdot (-4)+(-1) \cdot 5 = -9$$$.

A simple path from node $$$2$$$ to node $$$4$$$: $$$2 \rightarrow 1 \rightarrow 4$$$ has alternating function $$$A(2,4) = 1 \cdot (1)+(-1) \cdot (-4)+1 \cdot (-2) = 3$$$.

A simple path from node $$$1$$$ to node $$$1$$$ has a single node $$$1$$$, so $$$A(1,1) = 1 \cdot (-4) = -4$$$.

Similarly, $$$A(2, 1) = 5$$$, $$$A(3, 1) = 9$$$, $$$A(4, 2) = 3$$$, $$$A(1, 4) = -2$$$, $$$A(4, 1) = 2$$$, $$$A(2, 2) = 1$$$, $$$A(3, 3) = 5$$$, $$$A(4, 4) = -2$$$, $$$A(3, 4) = 7$$$, $$$A(4, 3) = 7$$$, $$$A(2, 3) = 10$$$, $$$A(3, 2) = 10$$$. So the answer is $$$(-5) + (-9) + 3 + (-4) + 5 + 9 + 3 + (-2) + 2 + 1 + 5 + (-2) + 7 + 7 + 10 + 10 = 40$$$.

Similarly $$$A(1,4)=-2, A(2,2)=1, A(2,1)=5, A(2,3)=10, A(3,3)=5, A(3,1)=9, A(3,2)=10, A(3,4)=7, A(4,4)=-2, A(4,1)=2, A(4,2)=3 , A(4,3)=7$$$ which sums upto 40.

F. Pathwalks

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a directed graph with n nodes and m edges, with all edges having a certain weight.

There might be multiple edges and self loops, and the graph can also be disconnected.

You need to choose a path (possibly passing through same vertices multiple times) in the graph such that the weights of the edges are in strictly increasing order, and these edges come in the order of input. Among all such paths, you need to find the the path that has the maximum possible number of edges, and report this value.

Please note that the edges picked don't have to be consecutive in the input.

Input

The first line contains two integers n and m $(1 \le n \le 100000, 1 \le m \le 100000)$ — the number of vertices and edges in the graph, respectively.

m lines follows.

The i-th of these lines contains three space separated integers a_i, b_i and w_i $(1 \le a_i, b_i \le n,\ 0 \le w_i \le 100000)$, denoting an edge from vertex a_i to vertex b_i having weight w_i

Output

Print one integer in a single line — the maximum number of edges in the path.

Examples

input

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

output

```
2
```

input

```
5 5
1 3 2
3 2 3
3 4 5
5 4 0
4 5 8
```

output

```
3
```

Note

The answer for the first sample input is 2: $1 \rightarrow 2 \rightarrow 3$. Note that you cannot traverse $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ because edge $3 \rightarrow 1$ appears earlier in the input than the other two edges and hence cannot be picked/traversed after either of the other two edges.

In the second sample, it's optimal to pick 1-st, 3-rd and 5-th edges to get the optimal answer: $1 \rightarrow 3 \rightarrow 4 \rightarrow 5$.

G. Bandit Blues

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

Japate, while traveling through the forest of Mala, saw N bags of gold lying in a row. Each bag has some **distinct** weight of gold between 1 to N. Japate can carry only one bag of gold with him, so he uses the following strategy to choose a bag.

Initially, he starts with an empty bag (zero weight). He considers the bags in some order. If the current bag has a higher weight than the bag in his hand, he picks the current bag.

Japate put the bags in some order. Japate realizes that he will pick A bags, if he starts picking bags from the front, and will pick B bags, if he starts picking bags from the back. By picking we mean replacing the bag in his hand with the current one.

Now he wonders how many permutations of bags are possible, in which he picks A bags from the front and B bags from back using the above strategy.

Since the answer can be very large, output it modulo 998244353.

Input

The only line of input contains three space separated integers N ($1 \le N \le 10^5$), A and B ($0 \le A, B \le N$).

Output

Output a single integer — the number of valid permutations modulo 998244353.

Examples

input
1 1 1
output
1

input
2 1 1
output
0

input
2 2 1
output
1

input
5 2 2
output
22

Note

In sample case 1, the only possible permutation is $[1]$

In sample cases 2 and 3, only two permutations of size 2 are possible: $\{[1, 2], [2, 1]\}$. The values of a and b for first permutation is 2 and 1, and for the second permutation these values are 1 and 2.

In sample case 4, out of 120 permutations of $[1, 2, 3, 4, 5]$ possible, only 22 satisfy the given constraints of a and b.

H. Santa's Gift

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Santa has an infinite number of candies for each of $$$m$$$ flavours. You are given a rooted tree with $$$n$$$ vertices. The root of the tree is the vertex $$$1$$$. Each vertex contains exactly one candy. The $$$i$$$-th vertex has a candy of flavour $$$f_i$$$.

Sometimes Santa fears that candies of flavour $$$k$$$ have melted. He chooses any vertex $$$x$$$ randomly and sends the subtree of $$$x$$$ to the Bakers for a replacement. In a replacement, all the candies with flavour $$$k$$$ are replaced with a new candy of the same flavour. The candies which are not of flavour $$$k$$$ are left unchanged. After the replacement, the tree is restored.

The *actual* cost of replacing one candy of flavour $$$k$$$ is $$$c_k$$$ (given for each $$$k$$$). The Baker keeps the price fixed in order to make calculation simple. Every time when a subtree comes for a replacement, the Baker charges $$$C$$$, no matter which subtree it is and which flavour it is.

Suppose that for a given flavour k the probability that Santa chooses a vertex for replacement is same for all the vertices. You need to find out the expected value of *error* in calculating the cost of replacement of flavour k. The error in calculating the cost is defined as follows.

$$Error\ E(k) =\ (Actual Cost\ –\ Price\ charged\ by\ the\ Bakers) ^2.$$

Note that the actual cost is the cost of replacement of one candy of the flavour k multiplied by the number of candies in the subtree.

Also, sometimes Santa may wish to replace a candy at vertex x with a candy of some flavour from his pocket.

You need to handle two types of operations:

- Change the flavour of the candy at vertex x to w.
- Calculate the expected value of error in calculating the cost of replacement for a given flavour k.

Input

The first line of the input contains four integers n ($2 \leqslant n \leqslant 5 \cdot 10^4$), m, q, C ($1 \leqslant m, q \leqslant 5 \cdot 10^4$, $0 \leqslant C \leqslant 10^6$) — the number of nodes, total number of different flavours of candies, the number of queries and the price charged by the Bakers for replacement, respectively.

The second line contains n integers $f_1, f_2, \dots, f_n$ ($1 \leqslant f_i \leqslant m$), where f_i is the initial flavour of the candy in the i-th node.

The third line contains $n - 1$ integers $p_2, p_3, \dots, p_n$ ($1 \leqslant p_i \leqslant n$), where p_i is the parent of the i-th node.

The next line contains m integers $c_1, c_2, \dots c_m$ ($1 \leqslant c_i \leqslant 10^2$), where c_i is the cost of replacing one candy of flavour i.

The next q lines describe the queries. Each line starts with an integer t ($1 \leqslant t \leqslant 2$) — the type of the query.

If $t = 1$, then the line describes a query of the first type. Two integers x and w follow ($1 \leqslant x \leqslant n$, $1 \leqslant w \leqslant m$), it means that Santa replaces the candy at vertex x with flavour w.

Otherwise, if $t = 2$, the line describes a query of the second type and an integer k ($1 \leqslant k \leqslant m$) follows, it means that you should print the expected value of the error in calculating the cost of replacement for a given flavour k.

The vertices are indexed from 1 to n. Vertex 1 is the root.

Output

Output the answer to each query of the second type in a separate line.

Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. The checker program considers your answer correct if and only if $\frac{|a-b|}{max(1,b)}\leqslant 10^{-6}$.

Example

input

```
3 5 5 7
3 1 4
1 1
73 1 48 85 89
2 1
2 3
1 2 3
2 1
2 3
```

output

```
2920.333333333333
593.000000000000
49.000000000000
3217.000000000000
```

Note

For 1-st query, the error in calculating the cost of replacement for flavour 1 if vertex 1, 2 or 3 is chosen are 66^2, 66^2 and $(-7)^2$ respectively. Since the probability of choosing any vertex is same, therefore the expected value of error is $\frac{66^2+66^2+(-7)^2}{3}$.

Similarly, for 2-nd query the expected value of error is $\frac{41^2+(-7)^2+(-7)^2}{3}$.

After 3-rd query, the flavour at vertex 2 changes from 1 to 3.

For 4-th query, the expected value of error is $\frac{(-7)^2+(-7)^2+(-7)^2}{3}$.

Similarly, for 5-th query, the expected value of error is $\frac{89^2+41^2+(-7)^2}{3}$.