

Swapping Brackets (bracketswap)

Do you remember Baby Bob? His older brother, Balanced Benjamin would like to give a gift to his brother. He has a string S of length N , where N is even. The string consists of $\frac{N}{2}$ opening brackets (“(”) and $\frac{N}{2}$ closing brackets (“)”). As Bob likes *valid bracket sequences*¹, Benjamin would like to turn S into a valid bracket sequence by swapping (not necessarily adjacent) characters in S .



Figure 1: Benjamin constructing the gift.

Since he’s less mathematically inclined than his younger brother, he asks for *your* help.

Write a program that turns S into a balanced bracket sequence by swapping characters in S using the minimum number of swaps. If there are more than one way to do this, you can output any of them.

 Among the attachments of this task you may find a template file `bracketswap.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integer N , the length of S .
- a line containing string S , a bracket sequence of length N .

Output

The output file consists of:

- a line containing integer R , the minimum number of swaps necessary.
- R lines, each line contains the indices (0-based) of the two characters to be swapped.

¹A bracket sequence is valid if “1” and “+” characters can be inserted into it so that it becomes a valid mathematical expression. For example, “()()()())” is a valid bracket sequence.

Constraints

- $1 \leq N \leq 1\,000\,000$.
- N is even.
- S has length N and only contains characters “(” and “)”.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
 
- **Subtask 2** (33 points) $N \leq 16$.
 
- **Subtask 3** (44 points) $N \leq 5678$.
 
- **Subtask 4** (23 points) No additional limitations.
 

Examples

input	output
4) (((	1 0 3
8))) (((((	2 0 5 7 1
10 () (() (()))	0

Explanation

In the **first sample case** the string after the swap looks like this: “(())”.

In the **second sample case** the swaps change string S the following way:

$$)))((((\Rightarrow ())((((\Rightarrow (((((($$

In the **third sample case** the bracket sequence is already valid.

Carry Bit (carry)

Valerio is currently visiting Poland, and on the street he found two strings A and B of length N each, consisting of 0's and 1's.



Figure 1: One of the strings found by Valerio.

When he returned home, he decided to write a program that chooses two contiguous substrings of length L , one from A and one from B , then interprets them as binary numbers (i.e. numbers written in base 2) and computes their sum. Unfortunately Valerio made a mistake: he decided to store the result of the sum in an L -bit variable (i.e., a variable that can store numbers from 0 to $2^L - 1$ inclusive), but in some cases the sum can be too large! Help him to verify the correctness of the result.

You are given Q queries, each consisting of 3 integers X , Y and L . For each query, determine whether Valerio's program is correct for the strings $A[X \dots X + L - 1]$ and $B[Y \dots Y + L - 1]$. In other words, you need to determine whether the sum of the numbers with binary representation $A[X \dots X + L - 1]$ and $B[Y \dots Y + L - 1]$ is strictly less than 2^L .

 Among the attachments of this task you may find a template file `carry.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integer N .
- a line containing string A of length N .
- a line containing string B of length N .
- a line containing integer Q .
- Q lines, the i -th of which consisting of integers X_i , Y_i and L_i .

Output

The output file must contain a single line consisting of the Q integers: the i -th value must be 1 if Valerio's program gives the correct result to query i , otherwise the value must be 0.

Constraints

- $1 \leq N \leq 200\,000$.
- A and B consist of characters '0' and '1'.
- $1 \leq Q \leq 200\,000$.
- $0 \leq X_i, Y_i < N$ for each $i = 0 \dots Q - 1$.
- $1 \leq X_i + L \leq N$ for each $i = 0 \dots Q - 1$.
- $1 \leq Y_i + L \leq N$ for each $i = 0 \dots Q - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
 
- **Subtask 2** (40 points) $N \leq 4000$ and $Q \leq 4000$.
 
- **Subtask 3** (30 points) $X_i = Y_i$ for each $i = 0 \dots Q - 1$.
 
- **Subtask 4** (30 points) No additional limitations.
 

Examples

input	output
7 1001010 0111101 5 0 0 1 0 0 3 0 0 4 0 4 3 4 5 2	1 1 0 0 1

input	output
12 101000010101 011110101110 7 1 5 4 8 2 3 9 4 2 10 3 1 0 0 3 0 0 6 3 5 6	1 0 0 1 0 0 1

Explanation

In the **first sample case** there are 5 queries:

- in the first query the 2 substrings are '1' and '0', and their sum is 1, less than 2^1 ;
- in the second query the 2 substrings are '100' and '011', and their sum is 7, less than 2^3 ;
- in the third query the 2 substrings are '1001' and '0111', and their sum is 16, not less than 2^4 ;
- in the fourth query the 2 substrings are '100' and '101', and their sum is 9, not less than 2^3 ;
- in the fifth query the 2 substrings are '01' and '01', and their sum is 2, less than 2^2 .

Dog Trick Competition (dogtrick)

RANDy is back at it again, but now he is ready to showcase his elite dog training skills. He's been training his dog Pompieru for many years, and now he is ready to participate in a dog trick competition.



Figure 1: Pompieru training for the competition.

There are $K + 1$ tricks which can be performed in this competition, numbered from 1 to $K + 1$. Pompieru does not know how to perform the trick $K + 1$, so he can only perform the tricks from 1 to K . Moreover, performing specific tricks one after the other is not always possible: he can only perform trick b right after trick a if he has been trained to do so. RANDy trained Pompieru to perform M pairs of consecutive tricks (A_i, B_i) , meaning that Pompieru knows how to perform trick B_i right after trick A_i .

The participants are required to perform N tricks $T_0, T_1, \dots, T_{N-1}$ in this specific order. For each successfully performed trick the dog gets 1 point. A dog may decide to skip a trick and continue with the next trick instead. The dogs are allowed to skip any number of tricks, even if they know how to do the tricks, but if a dog skips two consecutive tricks it gets disqualified, scoring zero points.

Help RANDy and Pompieru find the maximum number of points P they can score in this competition.

 Among the attachments of this task you may find a template file `dogtrick.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integers N and K .
- a line containing the N integers $T_0, \dots, T_{N-1}$.
- a line containing integer M .
- M lines, the i -th of which consisting of integers A_i, B_i .

Output

The output file must contain a single line consisting of integer P .

Constraints

- $2 \leq N \leq 200\,000$.
- $1 \leq K \leq 200\,000$.
- $1 \leq M \leq 200\,000$.
- $1 \leq T_i \leq K + 1$ for each $i = 0 \dots N - 1$.
- $1 \leq A_i, B_i \leq K$ for each $i = 0 \dots M - 1$.
- All the pairs (A_i, B_i) are distinct.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (15 points) $K = 1$.

- **Subtask 3** (19 points) $N, K, M \leq 20$.

- **Subtask 4** (34 points) $1 \leq N, M \leq 10\,000, 1 \leq K \leq 1000$.

- **Subtask 5** (32 points) No additional limitations.


Examples

input	output
6 3 4 1 1 2 2 2 2 2 2 1 2	4
4 3 1 4 2 3 1 1 3	0

Explanation

In the **first sample case**, Pompieru can skip the first trick, perform the second trick, skip the third and then perform the last three tricks, scoring 4 points total.

In the **second sample case**, Pompieru can perform the first trick but does not know how to perform the second trick, so he has to skip it. He can perform the third trick, but was not trained to perform it after the first trick, so he has to skip it again. Since he skipped two tricks in a row, he gets disqualified and scores 0 points.

Excellent Numbers 2 (excellent2)

A positive integer is called *excellent* if its decimal representation contains only the digits 1 and 5, and it is divisible by 3. For example, **15** and **111** are *excellent* numbers ($15 = 5 \cdot 3 + 0$ and $111 = 37 \cdot 3 + 0$), while **151** is not ($151 = 50 \cdot 3 + 1$).



Figure 1: 151515 is the hex triplet² of a variation of the *Eerie Black* color and also happens to be an *excellent* number!

Alex observed that there are many excellent numbers formed by N digits, and thus he started counting them. However, this is taking too much time, so he gave this task as a homework for you: help him count how many excellent numbers of N digits exist! Since the answer can be big, print it modulo $10^9 + 7$.

 Among the attachments of this task you may find a template file `excellent2.*` with a sample incomplete implementation.

Input

The first line of the input file contains a single integer T , the number of test cases. T test cases follow, each preceded by an empty line.

Each test case consists of:

- a line containing 64-bit integer N , representing the number of digits for which we have to find the answer.

²The *hex triplet* is a way to represent colors as its Red, Green, and Blue components with two hex digits (one byte) for each component.

Output

The output file must contain T lines corresponding to the test cases, each consisting of integer `ans`, representing the number of excellent numbers with N_i digits modulo $10^9 + 7$ (i.e. the remainder of the division by $10^9 + 7$).

✎ The *modulo* operation ($a \bmod m$) can be written in C/C++/Java/Python as `(a % m)`. To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!
Notice that if $x < 10^9 + 7$, then $2x$ fits into a C/C++/Java `int`.

Constraints

- $1 \leq T \leq 10$.
- $1 \leq N \leq 10^{18}$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
 
- **Subtask 2** (13 points) $N \leq 20$.
 
- **Subtask 3** (24 points) $N \leq 2000$.
 
- **Subtask 4** (34 points) $N \leq 200\,000$.
 
- **Subtask 5** (29 points) No additional limitations.
 

Examples

input	output
5	10
5	2
3	342
10	251936681
39	897205658
952	

Explanation

In the **first testcase** of the sample case, we have $N = 5$. There are 10 excellent numbers with 5 digits. In increasing order they are: 11115, 11151, 11511, 15111, 15555, 51111, 51555, 55155, 55515 and 55551.

In the **second testcase** we have $N = 3$. There are 2 excellent numbers with 3 digits: 115 and 555.

In the **fourth testcase** there are 183 251 937 962 excellent numbers with $N = 39$ digits. This number modulo $10^9 + 7$ is 251 936 681.

Stefan's New Year's resolutions (manymax)

Stefan has big plans for the year that just begun and among others, one of his plans is to improve his contest ratings. While every competitive programmer dreams of this, he is perfectly aware that not everyone can achieve it. Now he stumbled across an array that he received as a gift many years ago, together with a very old scroll containing the problem statement below.

You are given an array V of N positive integers, numbered from 0 to $N - 1$, inclusive. You have to answer Q queries: given L , R and K , find the **product** of the largest K values of $V_L, V_{L+1}, \dots, V_{R-1}, V_R$.

Stefan realized that this is not a cakewalk question so now it's your turn to help him overcome this challenge and write a program that answers the queries for him. Since the answers can be very big, you have to print them module $10^9 + 7$ (i.e. the remainder of the division by $10^9 + 7$).



Figure 1: Stefan discovering a long-lost problem.

📖 Among the attachments of this task you may find a template file `manymax.*` with a sample incomplete implementation.

📖 The *modulo* operation ($a \bmod m$) can be written in C/C++/Java/Python as `(a % m)`. To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!
Notice that if $x < 10^9 + 7$, then $2x$ fits into a C/C++/Java `int`.

Input

The input file consists of:

- a line containing integers N, Q .
- a line containing the N integers $V_0, \dots, V_{N-1}$.
- Q lines, the i -th of which consisting of integers L_i, R_i, K_i .

Output

The output file must contain a single line consisting of the Q integers `ans0, ..., ansQ-1`.

Constraints

- $1 \leq N \leq 200\,000$.
- $1 \leq V_i \leq 200\,000$ for each $i = 0 \dots N - 1$.
- $0 \leq L \leq R < N$ for each query.
- $1 \leq K \leq R - L + 1$ for each query.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
██████
- **Subtask 2** (11 points) $N, Q \leq 2000$.
██████
- **Subtask 3** (13 points) $K = R - L + 1$ for all queries.
██████
- **Subtask 4** (19 points) $K = 1$ for all queries.
██████
- **Subtask 5** (25 points) $N, Q \leq 50\,000$.
██████
- **Subtask 6** (32 points) No additional limitations.
██████

Examples

input	output
9 15 8 1 9 4 7 9 4 9 3 7 7 1 1 4 2 5 6 1 1 7 4 2 8 6 2 3 1 3 6 2 5 7 3 0 7 3 0 5 3 6 8 3 2 5 1 0 7 6 2 3 2 4 7 4	9 63 9 5103 81648 9 63 324 729 648 108 9 163296 36 2268

Explanation

For the **second query** of the sample case, the values between 1 and 4 are [1, 9, 4, 7], the 2 largest values are 9 and 7 and their product is 63.

For the **fourth query** of the sample case, the values between 1 and 7 are [1, 9, 4, 7, 9, 4, 9], the 4 largest values are 9, 9, 9 and 7 and their product is 5103.

Tuscan Peaks (peaks)

The mountains around Pisa are a well known hiking destination, with a peculiar terrain shape. In particular, you are interested in a rectangular area of $N \times M$ square meters, subdivided into $N \times M$ square cells, each with an area of a square meter. Those cells are uniquely identified by their distance from the top and the left sides of the rectangle, with cell (i, j) being at a distance of i meters from the top of the map and j meters from the left side of the rectangle.



Figure 1: Mount Point, the tallest mountain nearby Pisa.

Each cell (i, j) has an altitude of $H_{i,j}$ meters, described by two arrays of integers A and B : $H_{i,j} = A_i \cdot B_j$. A *peak* is a cell that has a strictly higher altitude than all the cells that share a side with it. How many *peaks* are there?

 Among the attachments of this task you may find a template file `peaks.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integers N, M .
- a line containing the N integers $A_0, \dots, A_{N-1}$.
- a line containing the M integers $B_0, \dots, B_{M-1}$.

Output

The output file must contain a single line consisting of 64-bit integer P , the number of peaks in the rectangular area.

Constraints

- $1 \leq N, M \leq 100\,000$.
- $1 \leq A_i \leq 10\,000$ for each $i = 0 \dots N - 1$.
- $1 \leq B_i \leq 10\,000$ for each $i = 0 \dots M - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- Subtask 1 (0 points) Examples.
 
- Subtask 2 (16 points) $N = 1$.
 
- Subtask 3 (33 points) $N, M \leq 1000$.
 
- Subtask 4 (51 points) No additional limitations.
 

Examples

input	output
1 5 7 4 6 8 5 1	1
4 5 3 2 8 4 8 5 6 1 3	6

Explanation

In the **first sample case**, the altitude of the cells is depicted below, with *peaks* highlighted in red.

28	42	56	35	7
----	----	----	----	---

In the **second sample case**, the altitude of the cells is depicted below, with *peaks* highlighted in red.

24	15	18	3	9
16	10	12	2	6
64	40	48	8	24
32	20	24	4	12

Strange Operation (strangeoperation)

Doctor Strange found two integer arrays of length N , namely $A_0, \dots, A_{N-1}$ and $B_0, \dots, B_{N-1}$. He can perform the following operation on A any number of times:

- Choose an index $1 \leq i \leq N - 2$ and let $A_i := -(A_{i-1} + A_i + A_{i+1})$. That is, assign the value $-(A_{i-1} + A_i + A_{i+1})$ to A_i .



Figure 1: Dr. Strange is preparing to perform some operations.

Help the Doctor determine whether it is possible to make the array A equal to B and if so, then find the minimum number of operations required to achieve this.

 Among the attachments of this task you may find a template file `strangeoperation.*` with a sample incomplete implementation.

Input

The first line contains the only integer N . The second line contains N integers, the elements of array A . The third line contains N integers, the elements of array B .

Output

You need to write a single line with an integer: the minimum number of operations required to make A equal to B , or -1 if it is not possible to do so.

Constraints

- $2 \leq N \leq 200\,000$.
- $-10^9 \leq A_i, B_i \leq 10^9$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

In this task, you can get **partial scores**: you will get 50% of the points for a subtask if you successfully determine whether it is possible to make A equal to B (but do not correctly solve all of its test cases). For this, the following condition must be satisfied for all test cases in a subtask: you should output -1 whenever it is impossible to make the two arrays equal, and otherwise, you should output a non-negative integer between 0 and $2^{63} - 1$.

- Subtask 1 (0 points)

Examples.

▢▢▢▢▢
- Subtask 2 (15 points)

There is at most one non-zero number in A .

▢▢▢▢▢
- Subtask 3 (18 points)

$N \leq 7$.

▢▢▢▢▢
- Subtask 4 (50 points)

$N \leq 1000$.

▢▢▢▢▢
- Subtask 5 (17 points)

No additional limitations.

▢▢▢▢▢

Examples

input	output
6 2 7 1 8 2 8 2 -10 1 -11 1 8	3
4 3 1 4 1 -4 1 -6 1	-1

Explanation

In the **first sample case**, consider the following steps for the array $A = [2, 7, 1, 8, 2, 8]$:

- Perform the operation on index 3. The array becomes: $[2, 7, 1, -11, 2, 8]$.
- Perform the operation on index 1. The array becomes: $[2, -10, 1, -11, 2, 8]$.
- Perform the operation on index 4. The array becomes: $[2, -10, 1, -11, 1, 8]$.

It is not possible to make the two arrays equal in less than 3 moves.

In the **second sample case**, it can be proven that there is no way to make A equal to B using the described operation.