Problem Set

Please check that you have 12 problems that are spanned across 22 pages in total (including this cover page).

A.	Apricot Seeds	(2 pages)
B.	Black Box	(2 pages)
C.	Farm	(2 pages)
D.	Fraction	(1 page)
E.	K-Lottery	(2 pages)
F.	Lucky Draws	(2 pages)
G.	Magic Cards	(2 pages)
H.	M. S. I. S.	(1 page)
I.	Product Delivery	(2 pages)
J.	Special Numbers	(1 page)
K.	Tandem Copy	(2 pages)
L.	Walk Swapping	(2 pages)

Problem A
Apricot Seeds
Time Limit: 3.0 Seconds

Sam has some apricot seeds, and he wants to sort them in non-decreasing order based on size. He uses a unique method to sort the apricot seeds, described as follows:

Given n apricot seeds, Sam performs a total of $n-1$ steps to sort them. For each step k from 1 to $n-1$:

- He compares the first seed with the second seed. If the second seed is smaller, he swaps their positions.
- He then compares the second seed with the third seed. If the third seed is smaller, he swaps their positions.
- He continues this process until he compares the $(n-k)$-th seed with the $(n-k+1)$-th seed and swaps their positions if the $(n-k+1)$-th seed is smaller.

Sam's friend Tom quickly realizes that this is the famous bubble sorting algorithm. To illustrate the inefficiency of this algorithm to Sam, Tom decides to ask Sam q questions. A question is represented as a tuple $[s,e,m,l,r]$. For given a sequence of n seeds, each question $[s,e,m,l,r]$ asks for the sum of the sizes of seeds from position l to r of the (partially) sorted subsequence after applying the first m steps of Sam's method to the subsequence of seeds from position s to e of the initial sequence.

For instance, consider four $(n=4)$ seeds with sizes of $(1,3,4,2)$ and two $(q=2)$ questions $[2,4,1,2,2]$ and $[1,4,2,3,4]$. For the first question, the subsequence of the sizes from the second $(s=2)$ seed to the fourth $(e=4)$ seed is $(3,4,2)$. After applying one step $(m=1)$ of Sam's method, it becomes $(3,2,4)$. The sum of the sizes of seeds from the second position $(l=2)$ to the second position $(r=2)$ in this (partially) sorted subsequence is 2. For the second question, the subsequence is $(1,3,4,2)$. After applying two steps, it becomes $(1,2,3,4)$. The sum of the sizes of seeds from position 3 to 4 in this (partially) sorted sequence is $3+4=7$.

Given a sequence of n seeds and q questions, write a program that computes the answer for each question.

Input
Your program is to read from standard input. The input starts with a line containing two integers, n and q $(2 \le n \le 1{,}000{,}000,\ 1 \le q \le 500{,}000)$, where n represents the number of seeds and q represents the number of questions. The second line contains n integers, separated by spaces, representing the sizes of the apricot seeds in their initial order. Each size is between 1 and 10^9, both inclusive. Each of the next q lines contains five positive integers s,e,m,l,r of query $[s,e,m,l,r]$, separated by spaces, representing a question, where $1 \le s < e \le n$, $1 \le m \le e-s$, and $1 \le l \le r \le e-s+1$.

Output
Your program is to write to standard output. For each of the q questions, output one line with the answer. The answer for a question $[s,e,m,l,r]$ is the sum of the sizes of seeds from position l to r of the partially sorted subsequence after applying the first m steps of Sam's method to the subsequence of seeds from position s to e of the input sequence.

The following shows sample input and output for three test cases.

Sample Input 1

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

Output for the Sample Input 1

```
2
7
```

Sample Input 2

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

Output for the Sample Input 2

```
1
5
3
```

Sample Input 3

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

Output for the Sample Input 3

```
6
11
```

Problem B
Black Box
Time Limit: 1.0 Seconds

The following **Python**-like pseudo code for function **BlackBox()** takes a list of positive integers and shuffles the integers in the list in a specific way, and returns the result as a list.

Three list methods are used below; For a list `L`, `len(L)` returns the number of items in `L`. `L.append(x)` adds the item `x` to the end of `L`. `L.pop(idx)` removes the item at the specified index `idx` from the list `L` and returns the removed item.

Given a list **Z** of positive integers, write a program to reconstruct a list **I** such that **Z = BlackBox(I)**.

```
function BlackBox( Banana ):
    if len( Banana ) <= 4 :
        exit("Too small Banana")
    Apple  = []                  # [] is an empty list
    Mango  = 0
    Papaya = len( Banana )

    while( Papaya >= 2 ) :
        Kiwi = Banana[ Mango ]
        Apple.append( Kiwi )
        Banana.pop( Mango )
        Papaya = Papaya - 1
        Mango = ( Kiwi + Mango - 1 ) % Papaya
    # end of while

    Apple.append( Banana[ 0 ] )
    Pear    = len( Apple ) - 1
    Orange  = Apple[ Pear ]
    Lime    = Apple[ 0 ]
    Coconut = Orange % Pear
    Melon   = Apple[ Coconut ]
    Apple[ 0 ] = Melon
    Apple[ Coconut ] = Lime

    return ( Apple )
# end of function BlackBox
```

Input

Your program is to read from standard input. The first line contains a positive integer n representing the number of positive integers of a list **`Z`**, where $5 \leq n \leq 200{,}000$. The following n lines contain n positive integers of the list **`Z`** returned from **`BlackBox(I)`**; the i-th line contains the i-th integer of the list **`Z`** between 1 and 100,000, both inclusive.

Output

Your program is to write to standard output. Print n integers of the list **`I`** where **`Z = BlackBox(I)`**, one per line; the i-th line should contain the i-th integer of **`I`**.

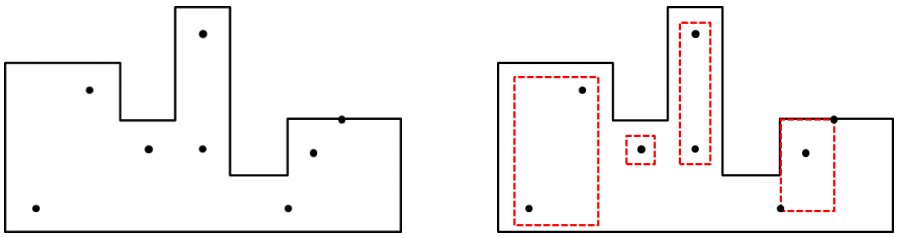
The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
13 113 49 68 91 10 179 2 71 78 45 57 10 88	10 113 179 68 57 45 10 2 88 71 49 78 91

Sample Input 2	Output for the Sample Input 2
9 6 8 7 9 5 1 2 4 3	9 8 7 6 5 1 2 3 4

Problem C
Farm
Time Limit: 0.7 Seconds

There is a farm that borders a straight road. Suppose the road is on the x-axis. Each boundary edge of the farm field is either horizontal or vertical. The leftmost and the rightmost edges are vertical and adjacent to the base edge which lies on the road. The length of the base edge is equal to the sum of the lengths of all other horizontal edges. See Figure C.1 (a).



(a) (b)

Figure C.1. A farm field and the pest infestation locations

In Figure C.1, the dots on the boundary or in the interior of the farm field represent the locations where the pests have infested. To effectively eradicate the infestation, a farmer tries to divide the infested area into several rectangular areas that satisfy the following conditions

- Each rectangular area must be contained within the farm. It is allowed for the edges of a rectangle to overlap the boundary of the farm.
- Each edge of a rectangular area is either horizontal or vertical.
- Rectangular areas are completely separated from each other, including their boundaries.
- Each pest infestation location must be contained within one of the rectangular areas. It is allowed for a pest infestation location to lie on an edge of a rectangle.

Figure C.1 (b) shows four rectangular areas covering all pest infestation locations. The farmer wants to minimize the number of rectangular areas for efficient pest management.

Given the boundary of a farm and the pest infestation locations, write a program to compute the minimum number of rectangular areas that satisfy the above conditions.

Input
Your program is to read from standard input. The input starts with a line containing two integers, m ($4 \le m \le 100{,}000$) and n ($0 \le n \le 100{,}000$), where m is the number of edges of the farm field and n is the number of the pest infestation locations. In the second line, m integers $v_1, v_2, \cdots, v_m$ ($v_1 = v_m = 0, 0 \le v_i \le 10^6$) are given, which represent the x-coordinates of the vertical edges and the y-coordinates of the horizontal edges. These vertical and horizontal edges are met alternately when traversing the upper boundary of the farm field clockwise from the left end of the base edge to the right end. From the third line, each of the n lines has two integers x and y, representing the coordinate (x, y) of a pest infestation location. All locations are on the boundary or in the interior of the farm field.

Output

Your program is to write to standard output. Print exactly one line. The line should contain an integer representing the minimum number of rectangular areas that satisfy the above conditions.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
12 8 0 30 20 20 30 40 40 10 50 20 70 0 4 5 15 26 25 15 35 15 35 35 50 5 55 15 60 20	4

Sample Input 2	Output for the Sample Input 2
4 0 0 10 50 0	0

Sample Input 3	Output for the Sample Input 3
12 3 0 3 2 6 4 1 6 4 8 2 10 0 3 5 7 3 3 1	2

Problem D
Fraction
Time Limit: 0.5 Seconds

A basic fraction can be represented by three integers $(a\ b\ c)$ which denotes $a + \frac{b}{c}$ where $1 \le a, b, c \le 9$. An extended fraction has the form of $(a'\ b'\ c')$ where a', b' and c' may be integers between one and nine or other extended fractions. Note that a basic fraction is also an extended fraction, and the length of the fraction is finite.

Given an extended fraction, we want to express its value as irreducible fraction. For example, the irreducible fraction of $\big((1\ 2\ 4)(5\ 2\ 3)(4\ 3\ (2\ 7\ 3))\big)$ is as follows.

$$\left(1 + \frac{2}{4}\right) + \frac{5 + \frac{2}{3}}{4 + \frac{3}{2 + \frac{7}{3}}} = \frac{991}{366}$$

Given a string form of an extended fraction, write a program that converts the extended fraction into the irreducible fraction.

Input
Your program is to read from standard input. The input starts with a line containing one integer n ($2 \le n \le 100$), where n is the number of symbols which are parentheses and digits between 1 and 9. The second line contains symbols, separated by a space, which represent an extended fraction.

Output
Your program is to write to standard output. Print exactly one line. If the answer is x/y, the line should contain two integers x and y, which are relatively prime to each other. Otherwise, (for example, when the input is not valid) print `-1`. You will need 64-bit integers to get the correct answer.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
5 (1 2 3)	5 3

Sample Input 2	Output for the Sample Input 2
8 (1 2 (3 4 5)	-1

Sample Input 3	Output for the Sample Input 3
21 ((1 2 4) (5 2 3) (4 3 (2 7 3)))	991 366

Problem E
K-Lottery
Time Limit: 2.0 Seconds

K-Lottery awards only one winner in each round. For each round, $K!$ tickets are produced and each ticket contains K different numbers from 1 to K, and no two tickets are identical. Among the tickets produced each round, M tickets are sold. The draw is conducted as follows each round. While randomly generating N ($N \geq K$) distinct numbers one by one, if the relative order of the last K consecutive numbers matches the numbers on any of the sold tickets, the draw ends immediately, and the corresponding ticket wins. Some rounds may not have any winning tickets.

For instance, let's consider a round where 6 tickets are produced ($K = 3$). The ticket sequences produced are $(1, 2, 3)$, $(1, 3, 2)$, $(2, 1, 3)$, $(2, 3, 1)$, $(3, 1, 2)$, and $(3, 2, 1)$. Among them, let's say $(1, 2, 3)$ and $(1, 3, 2)$ are sold ($M = 2$). Let's assume that the following 10 random numbers $(20, 35, 10, 7, 99, 53, 72, 33, 88, 16)$ are scheduled to be generated ($N = 10$). Then the relative order of $(7, 99, 53)$, say $(1, 3, 2)$ matches the sold ticket $(1, 3, 2)$, so that ticket wins.

In another scenario, let's consider a round where 24 tickets are produced ($K = 4$). The ticket sequences produced are $(1, 2, 3, 4)$, $(1, 2, 4, 3)$, $(1, 3, 2, 4)$, ..., and $(4, 3, 2, 1)$. Among them, let's assume $(1, 2, 3, 4)$, $(1, 2, 4, 3)$, $(3, 4, 1, 2)$, $(4, 1, 2, 3)$, and $(4, 2, 3, 1)$ are sold ($M = 5$). Let's assume that the following 10 random numbers $(19, 31, 9, 1, 89, 48, 63, 30, 78, 12)$ are scheduled to be generated ($N = 10$). Then the relative order of $(89, 48, 63, 30)$, say $(4, 2, 3, 1)$ matches the sold ticket $(4, 2, 3, 1)$, so that ticket wins.

Given information about a round of the K-Lottery, including the number of produced tickets, the number sequences of the sold tickets, and the sequence scheduled to be randomly generated for the winning ticket, write a program to output the number sequence of the winning ticket.

Input
Your program is to read from standard input. The input starts with a line containing three integers, K, M, and N ($3 \leq K \leq 10{,}000$, $1 \leq M \leq \min(K!, 1{,}000)$, $K \leq N \leq 1{,}000{,}000$, $3 \leq KM \leq 100{,}000$), where K is the number of numbers in each ticket, M is the number of tickets sold, and N is the number of numbers in the randomly generated sequence of the round. In each of the following M lines, K integers of a ticket sold in the round are given. The final line contains N different positive integers N_i ($1 \leq N_i \leq 100{,}000{,}000, 1 \leq i \leq N$) which is the number sequence for determining a winner.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the number sequence of the winning ticket. If there is no winning ticket, print `0`.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
3 2 10 1 2 3 1 3 2 20 35 10 7 99 53 72 33 88 16	1 3 2

Sample Input 2	Output for the Sample Input 2
4 5 10 1 2 3 4 1 2 4 3 3 4 1 2 4 1 2 3 4 2 3 1 19 31 9 1 89 48 63 30 78 12	4 2 3 1

Sample Input 3	Output for the Sample Input 3
3 3 7 1 3 2 2 3 1 2 1 3 11 22 33 44 55 66 77	0

Problem F
Lucky Draws
Time Limit: 2.0 Seconds

The ICPC committee is planning a surprise event to cheer on the participating teams. The committee provides each team with a pair of two numbers, A and B ($A \leq B$), before the competition, which will be used for the lucky draws after the competition. The committee wants to hold K draws. In each draw, a single number C is chosen by the committee, and all teams with a pair (A, B) such that $A \leq C \leq B$ win in this draw. To make more teams happy, the committee wants to choose the K numbers used in the K draws in advance so that the most teams win. A team can win multiple times but is considered to have won once.

For example, five teams are participating in ICPC and their pairs are $(1, 2), (1, 4), (3, 6), (4, 7), (5, 6)$, and $K = 2$. When the committee chooses two numbers 2 and 4, four teams with $(1, 2), (1, 4), (3, 6)$ and $(4, 7)$ win. The team with $(1, 4)$ wins twice because the pair contains both chosen numbers. In fact, all five teams can win if 2 and 5 are chosen. The maximum number of winning teams is five.

Given n pairs of two integers for teams and the number of lucky draws K, write a program to output the maximum number of winning teams.

Input
Your program is to read from standard input. The input starts with a line containing two integers, n and K ($1 \leq n \leq 10{,}000$, $1 \leq K \leq n$, $1 \leq n \times K \leq 500{,}000$), where n is the number of teams and K is the number of lucky draws. Each of the following n lines contains two integers A and B that represent the pair of a team, where $-10^6 \leq A \leq B \leq 10^6$.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the maximum number of winning teams. Teams that win more than once should only be counted once.

The following shows sample input and output for four test cases.

Sample Input 1

```
5 2
1 2
1 4
3 6
4 7
5 6
```

Output for the Sample Input 1

```
5
```

Sample Input 2

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

Output for the Sample Input 2

```
3
```

Sample Input 3	Output for the Sample Input 3
4 1 2 3 1 1 4 5 4 5	2

Sample Input 4	Output for the Sample Input 4
7 2 5 6 7 9 7 7 1 4 2 3 4 7 4 7	6

Problem G
Magic Cards
Time Limit: 3.0 Seconds

Chansu and Junsu are friends in International College of Programming Convergence. One day, Chansu met Junsu and said that "I'll do a magic trick for you. Pick any number between 1 and 12, and don't tell me your number. Just keep it in your mind." Junsu chose 11 in mind. Chansu then showed Junsu the following four cards one by one, asking "Is there your number in this card?" at each time.

1 9 7 11 3 5	2 10 3 6 7 11	4 5 6 7 6 12	8 11 10 9 12 9

So, Junsu answered "Yes, yes, no, yes" in this order. After Chansu did some magically looking gestures with his arms and legs for a while, he finally shouted, "I've got your number. It is 11." And Junsu was quite surprised because it was exactly the number he kept in mind.

Chansu didn't tell Junsu the secret of the trick, but only "These cards have a great magic power, so they can read your mind and tell me something only in a magical language, which only I can understand."

How does this work? Can you figure out the secret?

Now, you are to write a program that answers the number in your friends' minds. We can generalize the magic trick as follows: You have K magic cards in each of which exactly M integers between 1 and N, possibly with some redundancy, are written, and you perform the magic trick to F friends. From the yes/no-sequences from the F friends, you will be able to pick out the correct numbers.

Input
Your program is to read from standard input. The input starts with a line containing four integers, N, K, M, and F ($1 \le N \le 500{,}000$, $1 \le K \le 100$, $1 \le M \le 5{,}000$, $1 \le F \le 50{,}000$). In each of the following K lines, there are M integers between 1 and N, which represent the M numbers written in each magic card. In each of the following F lines, you are given a string of length K over {`Y`, `N`}, which represents the answer of each friend such that a `Y` means a "yes" and an `N` means a "no". You can assume that all the answers from the friends are correctly given according to their numbers chosen in mind.

Output
Your program is to write to standard output. Print exactly F lines. For each $i = 1,2,\dots,F$, the i-th line should consists of the number in the i-th friend's mind. If it is impossible to identify the one and only number of the i-th friend, print out `0` in the i-th line.

The following shows sample input and output for three test cases.

Sample Input 1	**Output for the Sample Input 1**
12 4 6 3 1 9 7 11 3 5 2 10 3 6 7 11 4 5 6 7 6 12 8 11 10 9 12 9 YYNY NNNY YNNN	11 8 1

Sample Input 2	**Output for the Sample Input 2**
13 4 6 4 1 9 7 11 3 5 2 10 3 6 7 11 4 5 6 7 6 12 8 11 10 9 12 9 YYNY NNNY YNNN NNNN	11 8 1 13

Sample Input 3	**Output for the Sample Input 3**
14 4 6 4 1 9 7 11 3 5 2 10 3 6 7 11 4 5 6 7 6 12 8 11 10 9 12 9 YYNY NNNY YNNN NNNN	11 8 1 0

Problem H
M. S. I. S.
Time Limit: 0.5 Seconds

We are given a $2 \times n$ matrix M of positive integers, and each row of M does not contain duplicate numbers. For i-th row r_i of M, $i = 1, 2$, we find the maximum sum s_i of increasing subsequence contained in r_i. For example, if M is given as the figure below, s_1 is $1 + 2 + 3 + 4 + 5 + 6$ and s_2 is $2 + 3 + 5$. We call $s_1 + s_2$ *the maximum sum of increasing subsequences, MSIS.*

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 3 & 5 & 4 & 1 \end{bmatrix}$$

Once we permute the columns of M, MSIS can change. For example, if we permute the columns of the above matrix $M = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6]$ to $[c_2\ c_3\ c_4\ c_5\ c_6\ c_1]$ as the figure below, MSIS becomes 36.

$$\begin{bmatrix} 2 & 3 & 4 & 5 & 6 & 1 \\ 2 & 3 & 5 & 4 & 1 & 6 \end{bmatrix}$$

Given a $2 \times n$ matrix M, write a program to output the maximum of MSIS among all possible permutations of the columns of M.

Input
Your program is to read from standard input. The input starts with a line containing an integer, n ($1 \le n \le 10{,}000$), where n is the number of columns of the input matrix M. In the following two lines, the i-th line contains n positive integers of the i-th row of M, for $i = 1, 2$. The integers given as input are between 1 and 50,000, and each row does not contain duplicate numbers.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the maximum of MSIS among all possible permutations of columns of M.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
6 1 2 3 4 5 6 6 2 3 5 4 1	36

Sample Input 2	Output for the Sample Input 2
5 50 40 3 2 1 1 2 3 100 200	396

Problem I
Product Delivery
Time Limit: 1.0 Seconds

There is only one railway line connecting $(n+1)$ cities developed along the coastline. When cities along the coast are sequentially identified by numbers between 0 and n, city $(i-1)$ and city i $(1 \leq i \leq n)$ are connected by rail, but other cities are not connected by rail.

Since every city except city 0 is famous as a tourist destination, every city $i(1 \leq i \leq n)$ excluding city 0 is preparing a variety of goods to welcome travelers ahead of the tourist season. Worldwide famous goods BFB is the most popular item in every city. However, the supplier of this product is located in city 0.

There is only one store that sells BFB in each city $i(1 \leq i \leq n)$. Let S_i be the BFB specialty store in city i. In each S_i, the number of BFBs expected to be sold in the tourist season is analyzed and reported to the supplier in the form of $[l_i, m_i]$. Here, l_i and m_i represent the minimum and the maximum number of expected required products, respectively.

The BFB supply company in city 0 collects request information from stores in every city and supplies products according to the rules described below.

- Select a city, say city $k(1 \leq k \leq n)$. Then, take a train departing from city 0, travel to city k, and supply BFBs only to the stores along the route. In other words, the BFB supplier supplies products to $S_1, S_2, \ldots, S_k$.
- Let c_i be the number of BFBs supplied to $S_i(1 \leq i \leq k)$ while moving along the route, the condition $c_i \leq c_{i+1}$ $(1 \leq i \leq k-1)$ must be satisfied.

If the supplier supplies products according to the supply rules described above, it may be impossible for every store to supply the desired number of items with a single supply procedure. Therefore, the supplier must go through several supply procedures to deliver the products but must comply with the supply rules described above each time. After completing all supply procedures, each S_i will have at least l_i and at most m_i items.

For example, suppose $n = 4$ and the number of items required by each store $S_i(1 \leq i \leq 4)$ are [13,15], [5,8], [6,14], and [3,7], respectively. In order for each store to supply the desired quantity of goods, there must be at least two delivery procedures. In the first delivery procedure, 6 items can be supplied to each of the 4 stores. Once delivery is completed in this first procedure, all stores' requests except S_1 are satisfied. Since 6 items have already been delivered to S_1, $r(7 \leq r \leq 9)$ additional products will be delivered to S_1 in the second delivery procedure. Of course, there may be other delivery methods. However, at least two delivery procedures are required.

Write a program to calculate the minimum number of supply procedures in order to supply the number of BFBs required by each store according to the above rules.

Input
Your program is to read from standard input. The input starts with a line containing an integer n $(1 \leq n \leq 10^6)$, where n is the number of cities in which the BFB specialty stores locate. In the following n lines, the i-th line contains two integers l_i and m_i $(1 \leq l_i \leq m_i \leq 10^9)$ which indicate the minimum and the maximum number of expected required products by S_i.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum number of supply processes in order to supply the number of products required by each store according to the delivery rules.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
4 13 15 5 8 6 14 3 7	2

Sample Input 2	Output for the Sample Input 2
5 1 2 2 3 33 44 4 5 6 7	2

Sample Input 3	Output for the Sample Input 3
5 10 20 3 6 13 30 7 8 11 13	3

Problem J
Special Numbers
Time Limit: 1.5 Seconds

Number theorist Dr. J is attracted by the beauty of numbers. When we are given a natural number $a = a_1a_2 \cdots a_n$ of n digits and a natural number k, a is called *k-special* if the product of all the digits of a, i.e. $a_1 \cdot a_2 \cdot a_3 \cdots a_n$ is divisible by k. Note that the number 0 is always divisible by a natural number.

For example, if $a = 2349$ and $k = 12$, then the product of all the digits of a, $2 \cdot 3 \cdot 4 \cdot 9 = 216$ is divisible by $k = 12$, so the number 2349 is 12-special. If $a = 2349$ and $k = 16$, then the product of all the digits of a, $2 \cdot 3 \cdot 4 \cdot 9 = 216$ is not divisible by $k = 16$, so the number 2349 is not 16-special.

Given three natural numbers k, L, and R, write a program to output z % $(10^9 + 7)$ where z is the number of k-special numbers among numbers in the range $[L, R]$.

Input
Your program is to read from standard input. The input has one line containing three integers, k, L, and $R(1 \leq k \leq 10^{17}, 1 \leq L \leq R \leq 10^{20})$.

Output
Your program is to write to standard output. Print exactly one line. The line should contain z % $(10^9 + 7)$ where z is the number of k-special numbers among the numbers in the range $[L, R]$, where both L and R are inclusive in the range.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
5 1 20	4

Sample Input 2	Output for the Sample Input 2
5 50 100	19

Sample Input 3	Output for the Sample Input 3
15 11 19	0

Problem K
Tandem Copy
Time Limit: 1.0 Seconds

Tandem copy is an operation on a DNA where a consecutive sequence of one or more nucleotides is repeated, and the repetitions are directly adjacent to each other; in other words, the tandem copy operation makes a copy of a consecutive sequence of nucleotides and pastes the copy right after the copied sequence. For example, $ATCATCG$ is resulted from the tandem copy of ATC in $ATCG$. Furthermore, we can continue another tandem copy on the resulted sequence $ATCATCG$ and obtain $ATCATTCG$. The following example illustrates a series of tandem copies from $ATCG$, where the underlined sequence is copied at each step.

$$\underline{ATC}G \Rightarrow ATCA\underline{T}CG \Rightarrow \underline{AT}CATTCG \Rightarrow ATATC\underline{ATT}CG \Rightarrow \cdots$$

We say that $ATCG$ produces all these sequences by tandem copy. It is easy to see that $ATCG$ can produce different sequences by selecting a different portion of the sequence to tandem copy at each step. Furthermore, in principle, $ATCG$ can produce infinitely many sequences by continuing tandem copies as many as it needs.

Usually, it is more expensive to tandem copy a longer portion. For instance,

$$\underline{ATC}G \Rightarrow ATCATCG$$

is a tandem copy of three nucleotides and thus is more expensive than

$$ATCA\underline{T}CG \Rightarrow ATCATTCG,$$

which is a tandem copy of one nucleotide. In other words, the length of a copied portion at each step is crucial to determine the tandem copy cost.

Since it is easy to tandem copy a single nucleotide, it is practical for the ICPC lab to store sequences such that two consecutive nucleotides in a sequence is always different; this helps the lab to reduce the storage space. For instance, since $ATTTG$ can be produced by tandem copying T twice from ATG, it is better for the lab to only store the shorter sequence ATG instead of $ATTTG$.

Because of a recent budget cut, the ICPC lab can only perform the tandem copy on at most two nucleotides at one time; namely, the length of a copied portion is at most two at each step. On the other hand, the lab can continue to repeat the tandem copy as many as it desires. For example, given a sequence $ABCD$, we can apply the tandem copy operation on B and obtain $ABBCD$, or apply it on the sequence BC and obtain $ABCBCD$. But we cannot tandem copy the consecutive sequence ABC because its length is longer than two.

Given a source string s and a target string t, your task is to count the number of all valid substrings s' of s, where one can obtain a string x from s' by applying an appropriate number of the tandem copy operations such that x contains t as a substring. Please note that no two consecutive nucleotides in the source string are the same, whereas two consecutive nucleotides in the target string can be the same. For example, CCA or $ATTGC$ cannot be source strings, but they can be target strings.

Now, given $s = ACATGCAT$ and $t = CCACATTT$, we take a substring $s' = CATGC$ of s and run a series of tandem copies as follows:

$$s' = \underline{C}ATGC \Rightarrow C\underline{CA}TGC \Rightarrow CCACA\underline{T}GC \Rightarrow CCACAT\underline{TG}C \Rightarrow CCACATTTGC,$$

which contains t as its substring.

Here is another substring example. For $s' = CAT$,

$$s' = \underline{CA}T \Rightarrow \underline{C}ACAT \Rightarrow CCACA\underline{T} \Rightarrow CCACAT\underline{T} \Rightarrow CCACATTT = t,$$

which shows that we can produce the target string from CAT by a series of tandem copies.

It is easy to verify that the total number of valid substrings of s is 14. Note that both the first CAT and the second CAT in s are counted as different valid substrings. Thus, you need to consider all substrings of s and count all valid substrings individually.

Here is another example. When $s = AB$ and $t = BA$, you can take the substring AB and tandem copy AB. Then, the resulting string is $ABAB$, which contains BA as its substring. All other substrings of s are unable to produce BA as a substring, and therefore the number of valid substrings is one.

Given a source string s and a target string t, where no two consecutive characters in s are the same character, write a program that outputs the number of valid substrings s' of s. s' is a valid substring of s if a series of tandem copies on s' can produce a string that contains t as its substring, where a tandem copy is restricted to at most two consecutive characters at each step.

Input

Your program is to read from standard input. The input consists of two lines. The first line is the source string s, and the second line is the target string t. Each input consists of uppercase letters A to Z, and $1 \leq |s|, |t| \leq 2 \times 10^4$.

Output

Your program is to write to standard output. Print exactly one line. The line should print the number of valid substrings in which a series of tandem copies can produce a string that contains the target string as its substring.

The following shows sample input and output for four test cases.

Sample Input 1	Output for the Sample Input 1
ACGCG CCG	9

Sample Input 2	Output for the Sample Input 2
TATCGC TTCCG	6

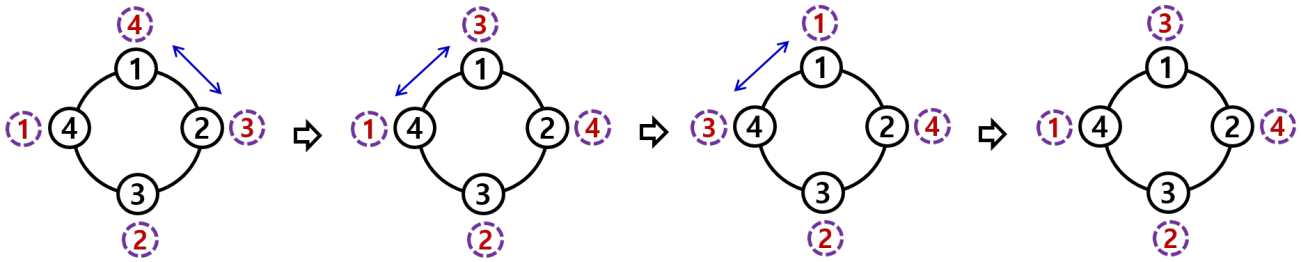
Sample Input 3	Output for the Sample Input 3
ABCABC ABC	7

Sample Input 4	Output for the Sample Input 4
ABCABC ABCABC	1

Problem L
Walk Swapping
Time Limit: 1.0 Seconds

A cycle C with n vertices is a graph such that the vertices i and $i+1$, for $i = 1, \dots, n-1$, are connected by an edge and also the vertices n and 1 are connected by an edge, where the vertices of C are numbered 1 to n.

There are n coins each of which is numbered as one of $\{1, 2, \dots, n\}$ and the numbers of two coins can be same. Initially, each vertex of C has a coin among those coins. Then two vertices can swap their coins with each other. For a walk $w = (v_1, v_2, \dots, v_k)$ in C, *a walk swapping* is to swap two coins on v_i and v_{i+1} in the order of $i = 1, 2, \dots, k-1$. Here, a walk w is a sequence of $k(\geq 1)$ vertices whose consecutive two vertices are different and adjacent in C, and it can be considered as the vertices visited when you traverse C. Also, k is called the length of w. For a walk of length $k \geq 2$, $k-1$ swaps in its walk swapping occur, and for a walk of length one, there is no swap. The figure below shows the progress of the walk swapping for the walk $(2, 1, 4, 1)$ in the cycle with 4 vertices.



There are given the initial configuration of coins that the vertices of C have in the first time and the final configuration of coins that the vertices should have in the end. You have to find a walk swapping that results in the final configuration of coins from the initial one, minimizing the total number of swaps of coins in the walk swapping.

For example, in the above figure, the number of swaps of coins in the walk swapping of the walk $(2, 1, 4, 1)$ is 3, however, the final configuration of coins is also achieved by the walk swapping of the walk $(1, 2)$ with only one swap.

Given the number of vertices of a cycle C and the initial and final configurations of coins on the vertices, write a program to output the minimum number of swaps of coins in a walk swapping resulting in the final configuration of coins from the initial one.

Input
Your program is to read from standard input. The input starts with a line containing one integer n ($1 \leq n \leq 3{,}000$), where n is the number of vertices in a cycle C. The vertices are numbered from 1 to n, and the coins on the vertices are numbered as ones of $\{1, 2, \dots, n\}$. The second line contains n integers, allowed for duplication, between 1 and n, where the i-th integer is the coin lying on the vertex i in the initial configuration of coins, for $i = 1, \dots, n$. The third line contains n integers, allowed for duplication, between 1 and n, where the i-th integer is the coin lying on the vertex i in the final configuration of coins, for $i = 1, \dots, n$.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum number of swaps of coins in a walk swapping that results in the final configuration of coins from the initial one. If there is no such walk swapping, that is, it is impossible to result in the final configuration by any walk swapping, print `-1`.

The following shows sample input and output for three test cases.

Sample Input 1

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

Output for the Sample Input 1

```
1
```

Sample Input 2

```
6
2 1 1 2 2 1
1 2 2 2 1 1
```

Output for the Sample Input 2

```
7
```

Sample Input 3

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

Output for the Sample Input 3

```
-1
```