Codeforces Round #493 (Div. 1)

A. Convert to Ones

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

You've got a string $$$a_1, a_2, \dots, a_n$$$, consisting of zeros and ones.

Let's call a sequence of consecutive elements $$$a_i, a_{i + 1}, \ldots, a_j$$$ ($$$1\leq i\leq j\leq n$$$) a *substring* of string $$$a$$$.

You can apply the following operations any number of times:

- Choose some substring of string $$$a$$$ (for example, you can choose entire string) and reverse it, paying $$$x$$$ coins for it (for example, «01011101» $$$\to$$$ «01110101»);
- Choose some substring of string $$$a$$$ (for example, you can choose entire string or just one symbol) and replace each symbol to the opposite one (zeros are replaced by ones, and ones — by zeros), paying $$$y$$$ coins for it (for example, «01011101» $$$\to$$$ «01100101»).

You can apply these operations in any order. It is allowed to apply the operations multiple times to the same substring.

What is the minimum number of coins you need to spend to get a string consisting only of ones?

Input

The first line of input contains integers $$$n$$$, $$$x$$$ and $$$y$$$ ($$$1 \leq n \leq 300\,000, 0 \leq x, y \leq 10^9$$$) — length of the string, cost of the first operation (substring reverse) and cost of the second operation (inverting all elements of substring).

The second line contains the string $$$a$$$ of length $$$n$$$, consisting of zeros and ones.

Output

Print a single integer — the minimum total cost of operations you need to spend to get a string consisting only of ones. Print $$$0$$$, if you do not need to perform any operations.

Examples

input
5 1 10 01000
output
11

input
5 10 1 01000
output
2

input
7 2 3 1111111
output
0

Note

In the first sample, at first you need to reverse substring $$$[1 \dots 2]$$$, and then you need to invert substring $$$[2 \dots 5]$$$.

Then the string was changed as follows:

«01000» $$$\to$$$ «10000» $$$\to$$$ «11111».

The total cost of operations is $$$1 + 10 = 11$$$.

In the second sample, at first you need to invert substring $$$[1 \dots 1]$$$, and then you need to invert substring $$$[3 \dots 5]$$$.

Then the string was changed as follows:

«01000» $$$\to$$$ «11000» $$$\to$$$ «11111».

The overall cost is $$$1 + 1 = 2$$$.

In the third example, string already consists only of ones, so the answer is $$$0$$$.

B. Roman Digits

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

Let's introduce a number system which is based on a roman digits. There are digits `I`, `V`, `X`, `L` which correspond to the numbers $$$1$$$, $$$5$$$, $$$10$$$ and $$$50$$$ respectively. The use of other roman digits is not allowed.

Numbers in this system are written as a sequence of one or more digits. We define the value of the sequence simply as the sum of digits in it.

For example, the number `XXXV` evaluates to $$$35$$$ and the number `IXI` — to $$$12$$$.

Pay attention to the difference to the traditional roman system — in our system any sequence of digits is valid, moreover the order of digits doesn't matter, for example `IX` means $$$11$$$, not $$$9$$$.

One can notice that this system is ambiguous, and some numbers can be written in many different ways. Your goal is to determine how many distinct integers can be represented by **exactly** $$$n$$$ roman digits `I`, `V`, `X`, `L`.

Input
The only line of the input file contains a single integer $$$n$$$ ($$$1 \le n \le 10^9$$$) — the number of roman digits to use.

Output
Output a single integer — the number of distinct integers which can be represented using $$$n$$$ roman digits `exactly`.

Examples

input
1
output
4

input
2
output
10

input
10
output
244

Note
In the first sample there are exactly $$$4$$$ integers which can be represented — `I`, `V`, `X` and `L`.

In the second sample it is possible to represent integers $$$2$$$ (`II`), $$$6$$$ (`VI`), $$$10$$$ (`VV`), $$$11$$$ (`XI`), $$$15$$$ (`XV`), $$$20$$$ (`XX`), $$$51$$$ (`IL`), $$$55$$$ (`VL`), $$$60$$$ (`XL`) and $$$100$$$ (`LL`).

C. Sky Full of Stars

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

On one of the planets of Solar system, in Atmosphere University, many students are fans of bingo game.

It is well known that one month on this planet consists of $$$n^2$$$ days, so calendars, represented as square matrix $$$n$$$ by $$$n$$$ are extremely popular.

Weather conditions are even more unusual. Due to the unique composition of the atmosphere, when interacting with sunlight, every day sky takes one of three colors: blue, green or red.

To play the bingo, you need to observe the sky for one month — after each day, its cell is painted with the color of the sky in that day, that is, blue, green or red.

At the end of the month, students examine the calendar. If at least one row or column contains only cells of one color, that month is called lucky.

Let's call two colorings of calendar different, if at least one cell has different colors in them. It is easy to see that there are $$$3^{n \cdot n}$$$ different colorings. How much of them are lucky? Since this number can be quite large, print it modulo $$$998244353$$$.

Input
The first and only line of input contains a single integer $$$n$$$ ($$$1 \le n \le 1000\,000$$$) — the number of rows and columns in the calendar.

Output
Print one number — number of lucky colorings of the calendar modulo $$$998244353$$$

Examples

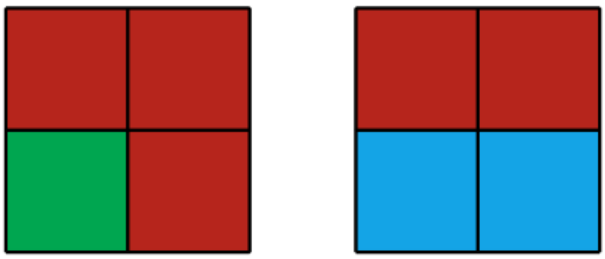
input
1
output
3

input
2
output
63

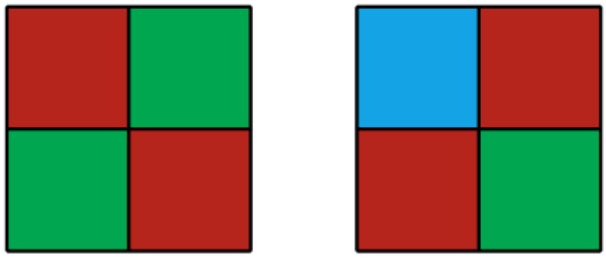
input
3
output
9933

Note
In the first sample any coloring is lucky, since the only column contains cells of only one color.

In the second sample, there are a lot of lucky colorings, in particular, the following colorings are lucky:



While these colorings are not lucky:



D. Cycles in product

time limit per test: 7 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider a tree (that is, an undirected connected graph without loops) $$$T_1$$$ and a tree $$$T_2$$$. Let's define their *cartesian product* $$$T_1 \times T_2$$$ in a following way.

Let $$$V$$$ be the set of vertices in $$$T_1$$$ and $$$U$$$ be the set of vertices in $$$T_2$$$.

Then the set of vertices of graph $$$T_1 \times T_2$$$ is $$$V \times U$$$, that is, a set of ordered pairs of vertices, where the first vertex in pair is from $$$V$$$ and the second — from $$$U$$$.

Let's draw the following edges:

- Between $$$(v, u_1)$$$ and $$$(v, u_2)$$$ there is an undirected edge, if $$$u_1$$$ and $$$u_2$$$ are adjacent in $$$U$$$.
- Similarly, between $$$(v_1, u)$$$ and $$$(v_2, u)$$$ there is an undirected edge, if $$$v_1$$$ and $$$v_2$$$ are adjacent in $$$V$$$.

Please see the notes section for the pictures of products of trees in the sample tests.

Let's examine the graph $$$T_1 \times T_2$$$. How much cycles (not necessarily simple) of length $$$k$$$ it contains? Since this number can be very large, print it modulo $$$998244353$$$.

The sequence of vertices $$$w_1$$$, $$$w_2$$$, ..., $$$w_k$$$, where $$$w_i \in V \times U$$$ called cycle, if any neighboring vertices are adjacent and $$$w_1$$$ is adjacent to $$$w_k$$$. Cycles that differ only by the cyclic shift or direction of traversal are still considered **different**.

Input

First line of input contains three integers — $$$n_1$$$, $$$n_2$$$ and $$$k$$$ ($$$2 \le n_1, n_2 \le 4000$$$, $$$2 \le k \le 75$$$) — number of vertices in the first tree, number of vertices in the second tree and the cycle length respectively.

Then follow $$$n_1 - 1$$$ lines describing the first tree. Each of this lines contains two integers — $$$v_i, u_i$$$ ($$$1 \le v_i, u_i \le n_1$$$), which define edges of the first tree.

Then follow $$$n_2 - 1$$$ lines, which describe the second tree in the same format.

It is guaranteed, that given graphs are trees.

Output

Print one integer — number of cycles modulo $$$998244353$$$.

Examples

input
2 2 2 1 2 1 2
output
8

input
2 2 4 1 2 1 2
output
32

input
2 3 4 1 2 1 2 1 3
output
70

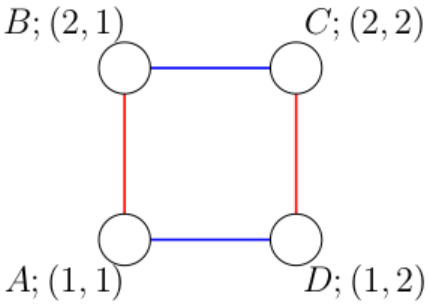
input
4 2 2 1 2 1 3 1 4 1 2
output
20

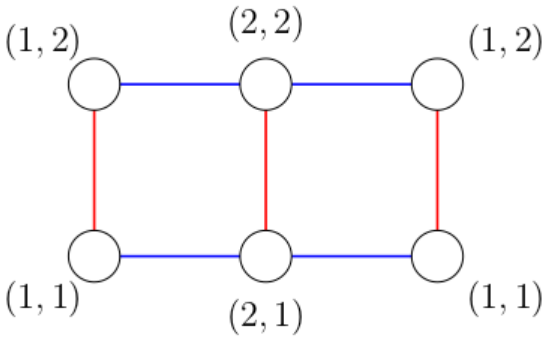
Note

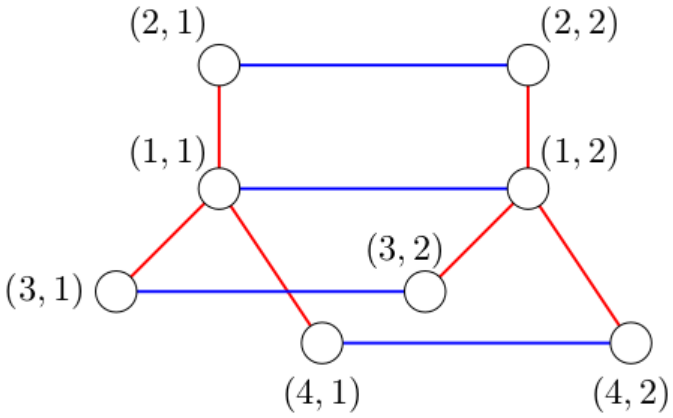
The following three pictures illustrate graph, which are products of the trees from sample tests.

In the first example, the list of cycles of length $$$2$$$ is as follows:

- «AB», «BA»
- «BC», «CB»
- «AD», «DA»
- «CD», «DC»







E. Good Subsegments

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A permutation $$$p$$$ of length $$$n$$$ is a sequence $$$p_1, p_2, \ldots, p_n$$$ consisting of $$$n$$$ distinct integers, each of which from $$$1$$$ to $$$n$$$ ($$$1 \leq p_i \leq n$$$) .

Let's call the subsegment $$$[l,r]$$$ of the permutation **good** if all numbers from the minimum on it to the maximum on this subsegment occur among the numbers $$$p_l, p_{l+1}, \ldots, p_r$$$.

For example, good segments of permutation $$$[1, 3, 2, 5, 4]$$$ are:

- $$$[1, 1]$$$,
- $$$[1, 3]$$$,
- $$$[1, 5]$$$,
- $$$[2, 2]$$$,
- $$$[2, 3]$$$,
- $$$[2, 5]$$$,
- $$$[3, 3]$$$,
- $$$[4, 4]$$$,
- $$$[4, 5]$$$,
- $$$[5, 5]$$$.

You are given a permutation $$$p_1, p_2, \ldots, p_n$$$.

You need to answer $$$q$$$ queries of the form: find the number of good subsegments of the given segment of permutation.

In other words, to answer one query, you need to calculate the number of good subsegments $$$[x \dots y]$$$ for some given segment $$$[l \dots r]$$$, such that $$$l \leq x \leq y \leq r$$$.

Input

The first line contains a single integer $$$n$$$ ($$$1 \leq n \leq 120000$$$) — the number of elements in the permutation.

The second line contains $$$n$$$ distinct integers $$$p_1, p_2, \ldots, p_n$$$ separated by spaces ($$$1 \leq p_i \leq n$$$).

The third line contains an integer $$$q$$$ ($$$1 \leq q \leq 120000$$$) — number of queries.

The following $$$q$$$ lines describe queries, each line contains a pair of integers $$$l$$$, $$$r$$$ separated by space ($$$1 \leq l \leq r \leq n$$$).

Output

Print a $$$q$$$ lines, $$$i$$$-th of them should contain a number of good subsegments of a segment, given in the $$$i$$$-th query.

Example

input

```
5
1 3 2 5 4
15
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
5 5
```

output

```
1
2
5
6
10
1
3
4
7
1
2
4
1
3
1
```