Problem A. Moving Dots

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

You are given n dots that are located on X-asis, i-th of which has a coordinate x_i. You can move dots. Furthermore, you know, that the dots are so small, that if they do intersect, they will merge into one dot. When two dots do merge, dot with smaller index overlap dot with bigger one, so its index will equal to the smallest index among their indexes. In one second, you can do the following:

1. Choose one dot among all current existing dots. Let its index be j.
2. Move dot with index j from position x_j to $(x_j - 1)$ or to $(x_j + 1)$.

In other words, in one second you can choose a dot and move either left or right by 1.

You are given q queries. For each query, you need to print the least number of dots that can remain on the plain after t seconds passed.

Input

The first line of input contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — number of dots.

The second line of input contain n integers x_i ($-10^6 \le x_i \le 10^6$) — coordinates of the dots on the line.

The next line of input contains one integer q ($1 \le q \le 2 \cdot 10^5$) — number of queries.

The following line of input contain q integers t_i ($0 \le t_i \le 10^9$).

Output

Print q lines — answers for the corresponding queries.

Scoring

1. (11 points): $n \le 3$;
2. (16 points): $0 \le x_i \le 20$;
3. (14 points): $t_i \le 20$;
4. (17 points): $n, q \le 3000$;
5. (18 points): $q \le 100$;
6. (24 points): no additional restrictions.

Examples

standard input	standard output
4 1 2 5 6 2 5 3	1 2
5 8 1 5 11 25 10 0 1 2 3 4 5 6 7 8 9	5 5 5 4 4 4 3 3 3 3

Problem B. Mex Permutations

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

Initially, there is an array a of n integers. It is known, that all its elements are in range $[0; n-1]$ and they are pairwise distinct. In other words, a is a permutation.

Let's denote $mex(S)$ the minimal non-negative integer that does not belong to the set S. For example, $mex(\{1,2,3\}) = 0$, $mex(\{0,1,3\}) = 2$, $mex(\{0,1,2\}) = 3$.

Array F is being built in such a way, that $F_i = mex(a_1, a_2, \ldots, a_{i-1}, a_i)$. Let's denote $f(a) = F$.

It is quite easy to build array F by given array a, but unfortunately, array a was lost. You wanted to restore it using array F, but there appeared to be too many possible arrays. You want to know how many arrays a exists, such that $f(a) = F$ modulo 998244353.

Input

The first line of input contains one integer n ($1 \le n \le 10^6$) — length of the permutation.

The second line of input contains n integers F_i ($0 \le F_i \le n+1$) — elements of array F.

Output

Print one integer — number of arrays a that satisfy $f(a) = F$ modulo 998244353.

Scoring

1. (3 points): $F_i = n + 1$;
2. (3 points): $F_i < F_{i+1}$ for all $i < n$;
3. (5 points): $n \le 3$;
4. (16 points): $n \le 7$;
5. (18 points): $n \le 18$;
6. (27 points): $n \le 1000$;
7. (28 points): no additional restrictions.

Examples

standard input	standard output
4 1 2 3 4	1
3 1 1 1	0
6 0 0 0 1 1 6	24

Problem C. Boring Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	256 megabytes

You are given a string s of length n. Let's define a function from a string:

$$f(s) = \sum_{i=1}^{n} \sum_{j=i}^{n} |s_i - s_j|$$

Here $|a - b|$ denotes a distance between characters a and b in alphabet. In other words, $f(s)$ denotes sum of $|s_i - s_j|$ over all $1 \le i \le j \le n$. Then, let

$$cost(s) = \sum_{i=1}^{n} \sum_{j=i}^{n} f(s[i..j])$$

Here $s[i..j]$ denotes substring of string s from index i to j. In other words, $cost(s)$ denotes sum of $f(s[l..r])$ over all $1 \le l \le r \le n$.

You are given q queries. Each query changes the element at position p to c, i.e., $s_p = c$. You need to print $cost(s)$ `mod` 1000000007 before all queries and after each query.

Input

The first line of input contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the string s.

The second line of input contains a string s of length n.

The third line of input contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

The following q lines contain an integer p ($1 \le p \le n$) and character c ($c \in \{a, b, \ldots, y, z\}$) — description of queries.

Output

Print $q + 1$ integers in separate lines — answers for the problem.

Scoring

1. (3 points): $n \le 3$;
2. (5 points): $n, q \le 50$;
3. (6 points): $n, q \le 100$;
4. (7 points): $c, s_i \in \{a, b\}$;
5. (20 points): $n, q \le 5000$;
6. (21 points): $n, q \le 20000$;
7. (13 points): $c, s_i \le$ "n";
8. (10 points): $n, q \le 10^5$;
9. (15 points): no additional restrictions.

Examples

standard input	standard output
3 aba 4 2 a 2 b 3 b 1 c	4 0 4 3 3
17 yuliiaalisadaryna 10 8 l 1 z 3 t 4 x 1 w 3 q 8 o 2 v 9 e 4 h	35140 35140 35276 36788 39884 39516 39132 39668 39824 40112 37072

Problem D. Graph? Are you sure?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

If you can't believe, you can't achieve.

Jeff Long

Initially, there are n vertexes and no edges. Each of the added edges has an integer c_i written on it. We define a simple path between two vertexes a and b, which belong to the same component, as the shortest path from a to b. Consider a simple path between two vertexes being *good* if each value on this path has the even number of entries. You have to answer q queries of 4 types:

- $1\ u\ v\ c$ $(1 \le c \le 4 \cdot 10^9)$ — add edge that connects vertexes u and v with value c on it
- $2\ u\ v$ — tell if the simple path between vertexes u and v is good. If there is no path between them, print -1
- $3\ u$ — tell the number of pairs of vertexes a and b $(1 \le a < b \le n)$, such that they belong to the same component with u and the path between them is considered good
- 4 — tell the number of pairs of vertexes a and b $(1 \le a < b \le n)$, such that the path between them is considered good, and there exists a path from a to b.

It is guaranteed that in all queries of the first type, the edge connects two different components.

Input

The first line of input contains two integers n $(1 \le n \le 2 \cdot 10^5)$ and q $(1 \le q \le 2 \cdot 10^5)$ — the number of vertices and queries accordingly.

Each of the following q lines contain a description of the query. First integer t_i defines the type of the query.

- $t_i = 1$ defines a query of the first type and is followed by 3 integers u, v $(1 \le u \ne v \le n)$, c $(1 \le c \le 4 \cdot 10^9)$
- $t_i = 2$ defines a query of the second type and is followed by 2 integers u, v $(1 \le u, v \le n)$
- $t_i = 3$ defines a query of the third type and is followed by 1 integer u $(1 \le u \le n)$
- $t_i = 4$ defines a query of the fourth type.

Output

For each query of type 2 print

- `-1`, if there is no existing simple path
- `YES`, if the simple path is considered *good*
- `NO` otherwise.

For each query of types 3 or 4 print one integer — answer for corresponding query.

You have to answer queries in order they appear in the input.

Scoring

1. (5 points): for all $t_i = 1$ $u_i = 1$;
2. (5 points): $n, q \leq 20$;
3. (7 points): $n, q \leq 1000$;
4. (3 points): $t_i \leq t_{i+1}$, $t_i \leq 2$, $c = 1$;
5. (6 points): $t_i \leq t_{i+1}$, $t_i \leq 2$, $c \leq 8$;
6. (11 points): $t_i \leq t_{i+1}$, $t_i \leq 2$;
7. (9 points): $q = n$, $t_i = 1$ for $1 \leq i < n$, $t_q = 4$;
8. (17 points): there exists integer e such that all $t_i = 1$ for all $1 \leq i \leq e$, and for all $e < j \leq q$ $t_i \neq 1$;
9. (9 points): $c \leq 1000$, graph and tests are generated randomly, i.e., for each vertex v ($2 \leq v \leq n$) we randomly choose p_v ($1 \leq p_v < v$), then we generate random permutation $perm$ and make $v = perm_v$, $p_v = perm_{p_v}$. Type of query is chosen randomly (we won't choose first type if there are no edges to add). All values in queries are picked randomly;
10. (28 points): no additional restrictions.

Example

standard input	standard output
5 11 1 1 2 1 2 1 3 1 2 3 1 3 2 2 1 2 1 2 4 2 1 2 5 2 3 3 2 4 5 2 1 4 4	-1 1 NO 2 YES NO 2