A. FizzBuzz Remixed

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   512 megabytes
```

FizzBuzz is one of the most well-known problems from coding interviews. In this problem, we will consider a remixed version of FizzBuzz:

Given an integer n, process all integers from 0 to n. For every integer such that its remainders modulo 3 and modulo 5 are the same (so, for every integer i such that $i \bmod 3 = i \bmod 5$), print `FizzBuzz`.

However, you don't have to solve it. Instead, given the integer n, you have to report how many times the correct solution to that problem will print `FizzBuzz`.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case contains one line consisting of one integer n ($0 \le n \le 10^9$).

Output

For each test case, print one integer — the number of times the correct solution will print `FizzBuzz` with the given value of n.

Standard Input	Standard Output
7 0 5 15 42 1337 17101997 998244353	1 3 4 9 270 3420402 199648872

Note

In the first test case, the solution will print `FizzBuzz` for the integer 0.

In the second test case, the solution will print `FizzBuzz` for the integers $0, 1, 2$.

In the third test case, the solution will print `FizzBuzz` for the integers $0, 1, 2, 15$.

,

B. Robot Program

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

There is a robot on the coordinate line. Initially, the robot is located at the point x ($x \neq 0$). The robot has a sequence of commands of length n consisting of characters, where L represents a move to the left by one unit (from point p to point $(p-1)$) and R represents a move to the right by one unit (from point p to point $(p+1)$).

The robot starts executing this sequence of commands (one command per second, in the order they are presented). However, whenever the robot reaches the point 0, the counter of executed commands is reset (i. e. it starts executing the entire sequence of commands from the very beginning). If the robot has completed all commands and is not at 0, it stops.

Your task is to calculate how many times the robot will enter the point 0 during the next k seconds.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of a test case contains three integers n, x and k ($1 \le n \le 2 \cdot 10^5$; $-n \le x \le n$; $n \le k \le 10^{18}$).

The second line of a test case contains a string s consisting of n characters L and/or R.

Additional constraint on the input: the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the number of times the robot will enter the point 0 during the next k seconds.

Standard Input	Standard Output
6 3 2 6 LLR 2 -1 8 RL 4 -2 5 LRRR 5 3 7 LRRLL 1 1 1 L 3 -1 4846549234412827 RLR	1 4 1 0 1 2423274617206414

Note

In the first example, the robot moves as follows: $2 \to 1 \to \underline{0} \to -1 \to -2 \to -1$. The robot has completed all instructions from the sequence and is not at 0. So it stops after 5 seconds and the point 0 is entered once.

In the second example, the robot moves as follows: $-1 \to \underline{0} \to 1 \to \underline{0} \to 1 \to \underline{0} \to 1 \to \underline{0} \to 1$. The robot worked 8 seconds and the point 0 is entered 4 times.

In the third example, the robot moves as follows: $-2 \to -3 \to -2 \to -1 \to \underline{0} \to -1$. The robot worked 5 seconds and the point 0 is entered once.

In the fourth example, the robot moves as follows: $3 \to 2 \to 3 \to 4 \to 3 \to 2$. The robot has completed all instructions from the sequence and is not at 0. So it stops after 5 seconds, without reaching the point 0.

,

C. Limited Repainting

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

You are given a strip, consisting of n cells, all cells are initially colored red.

In one operation, you can choose a segment of consecutive cells and paint them **blue**. Before painting, the chosen cells can be either red or blue. Note that it is not possible to paint them red. You are allowed to perform at most k operations (possibly zero).

For each cell, the desired color after all operations is specified: red or blue.

It is clear that it is not always possible to satisfy all requirements within k operations. Therefore, for each cell, a penalty is also specified, which is applied if the cell ends up the wrong color after all operations. For the i-th cell, the penalty is equal to a_i.

The penalty of the final painting is calculated as the **maximum penalty** among all cells that are painted the wrong color. If there are no such cells, the painting penalty is equal to 0.

What is the minimum penalty of the final painting that can be achieved?

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n \le 3 \cdot 10^5$; $0 \le k \le n$) — the length of the strip and the maximum number of operations.

The second line contains a string s, consisting of n characters 'R' and/or 'B'. 'R' means that the cell should be painted red. 'B' means that the cell should be painted blue.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the penalty for each cell.

The sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print a single integer — the minimum penalty of the final painting.

Standard Input	Standard Output
5 4 1 BRBR 9 3 5 4 4 1 BRBR 9 5 3 4 4 2 BRBR 9 3 5 4 10 2	3 3 0 4 0

BRBRBBRRBR 5 1 2 4 5 3 6 1 5 4 5 5 RRRRR 5 3 1 2 4	

Note

In the first test case, you can paint the cells from 1 to 3. The painting will be BBBR. So, only cell 2 is painted the wrong color. Therefore, the penalty for it is the final penalty and equals 3.

In the second test case, the painting BBBR will now result in a penalty of 5. However, if you paint the cells from 1 to 1, resulting in BRRR, then only cell 3 is painted the wrong color. The final penalty is 3.

In the third test case, you can paint the cells from 1 to 1 and from 3 to 3. Then all cells will be the correct color, the penalty equals 0.

,

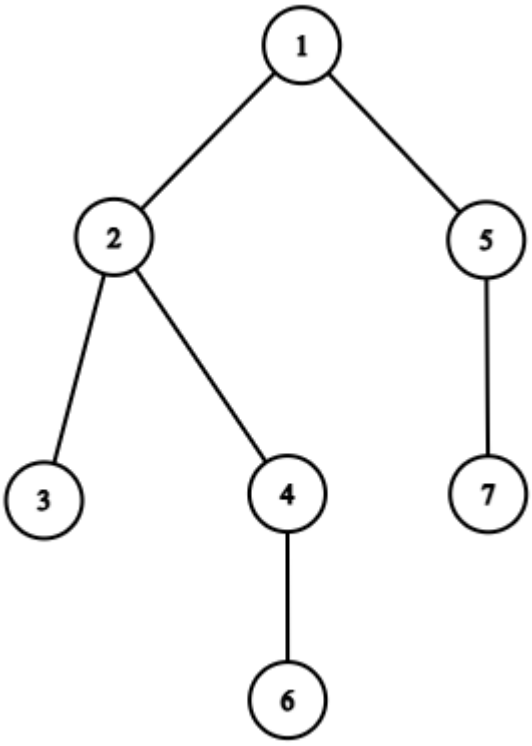
D. Tree Jumps

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

You are given a rooted tree, consisting of n vertices. The vertices in the tree are numbered from 1 to n, and the root is the vertex 1. Let d_x be the distance (the number of edges on the shortest path) from the root to the vertex x.

There is a chip that is initially placed at the root. You can perform the following operation as many times as you want (possibly zero):

- move the chip from the current vertex v to a vertex u such that $d_u = d_v + 1$. If v is the root, you can choose any vertex u meeting this constraint; however, if v is not the root, u should not be a neighbor of v (there should be no edge connecting v and u).



For example, in the tree above, the following chip moves are possible: $1 \to 2$, $1 \to 5$, $2 \to 7$, $5 \to 3$, $5 \to 4$, $3 \to 6$, $7 \to 6$.

A sequence of vertices is **valid** if you can move the chip in such a way that it visits all vertices from the sequence (and only them), in the order they are given in the sequence.

Your task is to calculate the number of valid vertex sequences. Since the answer might be large, print it modulo 998244353.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \le n \le 3 \cdot 10^5$).

The second line contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i is the parent of the i-th vertex in the tree. Vertex 1 is the root.

Additional constraint on the input: the sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case, print a single integer — the number of valid vertex sequences, taken modulo 998244353.

Standard Input	Standard Output
3 4 1 2 1 3 1 2 7 1 2 2 1 4 5	4 2 8

Note

In the first example, the following sequences are valid: $[1]$, $[1, 2]$, $[1, 4]$, $[1, 4, 3]$.

In the second example, the following sequences are valid: $[1]$, $[1, 2]$.

In the third example, the following sequences are valid: $[1]$, $[1, 2]$, $[1, 2, 7]$, $[1, 2, 7, 6]$, $[1, 5]$, $[1, 5, 3]$, $[1, 5, 3, 6]$, $[1, 5, 4]$.

,

E. Game with Binary String

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   512 megabytes
```

Consider the following game. Two players have a binary string (a string consisting of characters `0` and/or `1`). The players take turns, the first player makes the first turn. During a player's turn, he or she has to choose exactly two adjacent elements of the string and remove them (**the first element and the last element are also considered adjacent**). Furthermore, there are additional constraints depending on who makes the move:

- if it's the first player's move, **both** chosen characters should be `0`;
- if it's the second player's move, **at least one** of the chosen characters should be `1`.

The player who can't make a valid move loses the game. This also means that if the string currently has less than 2 characters, the current player loses the game.

You are given a binary string s of length n. You have to calculate the number of its substrings such that, if the game is played on that substring and both players make optimal decisions, the first player wins. In other words, calculate the number of pairs (l, r) such that $1 \le l \le r \le n$ and the first player has a winning strategy on the string $s_l s_{l+1} \ldots s_r$.

Input

The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$).

The second line contains the string s, consisting of exactly n characters. Each character of the string is either `0` or `1`.

Output

Print one integer — the number of substrings such that, if the game is played on that substring, the first player wins.

Standard Input	Standard Output
10 0010010011	12

Note

In the first example, the following substrings are winning for the first player ($s[l : r]$ denotes $s_l s_{l+1} \ldots s_r$):

- $s[1 : 2]$;
- $s[1 : 3]$;
- $s[1 : 7]$;
- $s[2 : 4]$;
- $s[2 : 8]$;
- $s[3 : 5]$;
- $s[4 : 5]$;
- $s[4 : 6]$;
- $s[5 : 7]$;
- $s[6 : 8]$;
- $s[7 : 8]$;

- $s[7:9]$.

,

F. Friends and Pizza

Input file: standard input
Output file: standard output
Time limit: 8 seconds
Memory limit: 512 megabytes

Monocarp has n pizzas, the i-th pizza consists of a_i slices. Pizzas are denoted by uppercase Latin letters from A to the n-th letter of the Latin alphabet.

Monocarp also has m friends, and he wants to invite **exactly two** of them to eat pizza. For each friend, Monocarp knows which pizzas that friend likes.

After the friends arrive at Monocarp's house, for each pizza, the following happens:

- if the pizza is not liked by any of the two invited friends, Monocarp eats it;
- if the pizza is liked by exactly one of the two invited friends, that friend eats it;
- and if the pizza is liked by both friends, they try to split it. If it consists of an even number of slices, they both eat exactly half of the slices. But if the pizza consists of an odd number of slices, they start quarrelling, trying to decide who will eat an extra slice — and Monocarp doesn't like that.

For each k from 0 to $\sum a_i$, calculate the number of ways to choose **exactly two friends** to invite so that the friends don't quarrel, and Monocarp eats exactly k slices.

Input

The first line contains two integers n and m ($1 \le n \le 20$; $2 \le m \le 5 \cdot 10^5$) — the number of pizzas and the number of friends, respectively.

The second line contains m strings $s_1, s_2, \ldots, s_m$ ($1 \le |s_i| \le n$), where s_i is a string consisting of **distinct** characters from A to the n-th letter of the Latin alphabet, denoting which pizzas the i-th friend likes. In every string s_i, the characters are sorted in lexicographical (alphabetic) order.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot 10^4$) — the sizes of the pizzas.

Output

Print $\sum a_i + 1$ integers, where the k-th integer (starting from 0) should be the number of ways to choose **exactly two friends** to invite so that the friends don't quarrel, and Monocarp eats exactly k slices.

Standard Input	Standard Output
3 6 A AB ABC AB BC C 2 3 5	4 0 0 1 0 2 0 0 0 0 0

Note

Let's consider all pairs of friends from the first example:

- if Monocarp invites friends 1 and 2, they will eat pizzas 1 and 2, and he'll eat the 3-rd pizza;
- if Monocarp invites friends 1 and 3, they will eat all of the pizzas;
- if Monocarp invites friends 1 and 4, they will eat pizzas 1 and 2, and he'll eat the 3-rd pizza;
- if Monocarp invites friends 1 and 5, they will eat all of the pizzas;
- if Monocarp invites friends 1 and 6, they will eat pizzas 1 and 3, and he'll eat the 2-nd pizza;

- if Monocarp invites friends 2 and 3, they will quarrel because of the 2-nd pizza;
- if Monocarp invites friends 2 and 4, they will quarrel because of the 2-nd pizza;
- if Monocarp invites friends 2 and 5, they will quarrel because of the 2-nd pizza;
- if Monocarp invites friends 2 and 6, they will eat all of the pizzas;
- if Monocarp invites friends 3 and 4, they will quarrel because of the 2-nd pizza;
- if Monocarp invites friends 3 and 5, they will quarrel because of the 2-nd pizza;
- if Monocarp invites friends 3 and 6, they will quarrel because of the 3-rd pizza;
- if Monocarp invites friends 4 and 5, they will quarrel because of the 2-nd pizza;
- if Monocarp invites friends 4 and 6, they will eat all of the pizzas;
- if Monocarp invites friends 5 and 6, they will quarrel because of the 3-rd pizza.