Youth Finale

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

Finales are born to be exciting. Performers play hard to draw audiences' attention and then take a perfect curtain call. As the last problem and the finale of the problem set, however, we want you to recall a simple algorithm. Like me, it may be the first algorithm you've learned, called Bubble Sort.

```
void bubble_sort(int a[], int n) { // 0-based, sort from lowest to highest
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < n - i; j++) {
            if (a[j] > a[j + 1]) {
                swap(a[j], a[j + 1]);
            }
        } // after i-th inner iteration, a[n - i] is correct
    }
}
```

Given a permutation of length n, as you might know, Bubble Sort runs in $\Omega(n^2)$ in the worst case. It's quite a traditional idea to count the number of calls of "`swap`" in the algorithm. As you are stronger now, you want to count that number in a dynamic permutation with the following events that might happen:

- **Reverse** the permutation, meaning that the permutation is replaced with

$$p' = \{p_n, p_{n-1}, \ldots, p_2, p_1\}.$$

- **Shift** the permutation to the left by 1, meaning that the permutation is replaced with

$$p' = \{p_2, p_3, \ldots, p_n, p_1\}.$$

All you need to do is to output the number of "`swap`" that would be called if we sort the permutation with the above Bubble Sort code after each operation.

Input

The first line contains two integers $n, m (1 \le n \le 3 \times 10^5, 1 \le m \le 6 \times 10^5)$, denoting the length of permutation and the number of operations.

The second line contains n integers separated by spaces, and the i-th denotes the initial p_i.

The third line contains a single string containing m letters consisting of '`R`' and '`S`'. The i-th letter denotes the i-th operation, where '`R`' or '`S`' denotes the **Reverse** or **Shift** operation, respectively.

It's guaranteed that p forms a correct permutation of $1, 2, \ldots, n$.

Output

In the first line, print the number of "`swap`" would be called when Bubble Sort the initial p.

In the second line, print a single string of m digits. The i-th denotes the number of "`swap`" would be called to Bubble Sort the permutation, **modulo** 10.

Example

standard input	standard output
5 10 5 4 3 2 1 SSSSRSSSSR	10 6446466400