Problem B
Between Strings

Today, in an algorithms course, Hanh learns lexicographical order.

The professor gives Hanh a counting task: *Given two strings a and b and an integer l, count the number of strings c of length l such that c is greater than a and less than b, in terms of lexicographical order.*

This task is quite a simple one for Hanh. He manages to write the below function in a matter of second: `countBetween(a, b, l, p)`. This function works as below:

- It has four parameters: `a` and `b` are two strings of lowercase English letters, `l` and `p` are two positive integers.
- It correctly counts the number of strings c containing exactly `l` lowercase English letters, satisfying $a < c < b$ in terms of lexicographical order.
- It returns a non-negative integer which is the remainder after dividing the number of such strings c by `p`.

To make thing more interesting and challenging for Hanh, his professor now asks him to proceed multiple queries of counting strings between. In order to solve this harder problem, Hanh writes the following pseudo code:

```
function solve(s, w, d, p):
    a = "icpcvn"
    b = "icpcvn"
    res = 0

    for each i from 0 to length(s) - 1:
        if w[i] equals 'A':
            insert character s[i] at the end of string a
        else:
            insert character s[i] at the end of string b

        sum = 0
        for each integer l in d:
            sum += countBetween(a, b, l, p)

        res = res xor (sum % p * (i + 1))

    return res
```

The function `solve` takes four parameters:

- Two strings `s` and `w` **of the same length**, `s` contains lowercase English characters only, `w` contains characters `A` and `B` only.
- `d` is a sequence of positive integers.

- `p` is a positive integer.

In the above pseudo code, `length(s)` denotes the number of characters of string `s`, while `s[i]` denotes the `i`-th character of string `s`. Characters are numbered starting from 0.

Sadly, Hanh realizes that his code may run too slow with some of the professor's test data. Hence, he asks you to write a program to efficiently calculate the returning value of the above function, given the values of all its parameters.

As a reminder, a string $X = x_0x_1x_2 \dots x_m$ is considered *lexicographically less* than a string $Y = y_0y_1y_2 \dots y_n$ iff either of the below conditions hold:

- $m < n$ and $x_i = y_i$ for every $0 \le i \le m$
- There exists an index i such that $i \le \min(m, n)$, $x_i < y_i$ and $x_j = y_j$ for every $0 \le j < i$.

A string X is considered *lexicographically greater* than a string Y iff Y is *lexicographically less* than X.

Input

- The first line contains three integers q, n and p $(1 \le q \le 2 \cdot 10^6, 1 \le n \le 7 \cdot 10^4, 1 \le p \le 10^9)$ — the length of parameters `s` and `w`, the number of elements of parameter `d` and the value of parameter `p`, respectively.
- The second line contains n integers $d_1, d_2, \dots, d_n$ $(1 \le d_i \le 10^9)$ — the elements of parameter `d`.
- The third line contains a string of exactly q lowercase English characters — the parameter `s`.
- The forth line contains a string of exactly q characters `A` and `B` — the parameter `w`.

Output

Print a single integer — the returned value of the above function `solve`.

Sample Explanation

In the above example, `s = "ca"`, `w = "BA"`, `d = [6, 7, 8]` and `p = 45`. In the beginning, `a = b = "icpcvn"`.

When `i = 0`, `a = "icpcvn"` and `b = "icpcvnc"`.

- `countBetween("icpcvn", "icpcvnc", 6, 45)` returns 0.
- `countBetween("icpcvn", "icpcvnc", 7, 45)` returns 2 as there are 2 strings of length 7 between `icpcvn` and `icpcvnc`: `icpcvna` and `icpcvnb`.
- `countBetween("icpcvn", "icpcvnc", 8, 45)` returns $52 \mod 45 = 7$ as there are 52 strings of length 8 between `icpcvn` and `icpcvnc`: `icpcvnaa`, `icpcvnab`, . . ., `icpcvnaz`, `icpcvnba`, `icpcvnbb`, . . ., `icpcvnbz`.

When `i = 1`, `a = "icpcvna"` and `b = "icpcvnc"`.

- `countBetween("icpcvna", "icpcvnc", 6, 45)` returns 0.
- `countBetween("icpcvna", "icpcvnc", 7, 45)` returns 1 as there is 1 string of length 7 between `icpcvna` and `icpcvnc`: `icpcvnb`.
- `countBetween("icpcvna", "icpcvnc", 8, 45)` returns $52 \mod 45 = 7$ as there are 52 strings of length 8 between `icpcvna` and `icpcvnc`: `icpcvnaa`, `icpcvnab`, . . ., `icpcvnaz`, `icpcvnba`, `icpcvnbb`, . . ., `icpcvnbz`.

Sample Input 1

```
2 3 45
6 7 8
ca
BA
```

Sample Output 1

```
25
```