Infinitely Finite

This problem was worth 500 points and was ranked easy. The authors of this problem are Aryan V S and Dhruv Chawla.

Note: GitHub does not support LaTex in Markdown. If you want a more readable version of the problem, download the PDF file instead.

Statement

Dhruv gives Aryan a string s of length n. He asks Aryan to perform the following operations:

- Assign $t = s$
- Assign $r = reverse(s)$

Once Aryan completes the above operations, Dhruv asks him to perform the following operation an infinite number of times:

- $s = s + r + t$

The $reverse(s)$ operation returns the reverse of a string i.e. "abcd" becomes "dcba", and "abc" becomes "cba", etc.

The $x + y$ operation is the concatenation of two strings x and y.

Dhruv then asks Aryan q questions. Each question consists of an integer a_i and is of the form "What is the character present at s_{a_i}?". Aryan thinks that the problem is trivial and asks you to solve it instead.

Note: The problem uses 1-based indexing in the testcases ($1 \leq a_i \leq 10^{18}$).

Input Format

The first line of the input contains a single integer n - the length of the string s initially.

The second line contains the string s - the string contains only lowercase English characters a - z.

The third line contains a single integer q - the number of questions.

The fourth line contains q space separated integers a_i - the integers corresponding to the questions.

Output Format

The output should contain q lines.

The i^{th} line should contain the answer to the i^{th} question - the character present at s_{a_i}.

Constraints

$1 \leq n \leq 2 * 10^5$

$1 \leq q \leq 5 * 10^5$

$1 \leq a_i \leq 10^{18}$

Sample Tests

Sample Test 1

Input

```
5
clown
5
15 14 3 9 1000000000000000000
```

Output

```
n
w
o
l
c
```

Solution

Time Complexity: $O(n+q)$

Space Complexity: $O(n)$

After performing the given operations on the input string "clown", we get something like: $s =$ *clownnwolcclownnwolcclown...*

We can notice that every $2n$ characters, the string repeats itself, and therefore we only need to care about the first $2n$ characters of s. That is, the i^{th} character is the same as the $(2n + i^{th})$ character and the $(4n + i^{th})$ character and so on. We can take all those indices modulo $2n$ and print the character corresponding to that position.

Solution in C++:

```cpp
#include <bits/stdc++.h>

int main () {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int n, q;
  std::string s;
  std::cin >> n >> s >> q;

  while (q--) {
    int64_t a;
    std::cin >> a;
    --a;

    int64_t parity = a / n;

    if (parity & 1)
      std::cout << s[n - 1 - a % n] << '\n';
    else
      std::cout << s[a % n] << '\n';
  }

  return 0;
}
```

Solution in Python:

```python
n = int(input())
s = input()
q = int(input())
queries = list(map(int, input().split()))

s = s + s[::-1]

for i in range(q):
  index = queries[i] - 1
  print(s[index % (2 * n)])
```