Linked List Loop

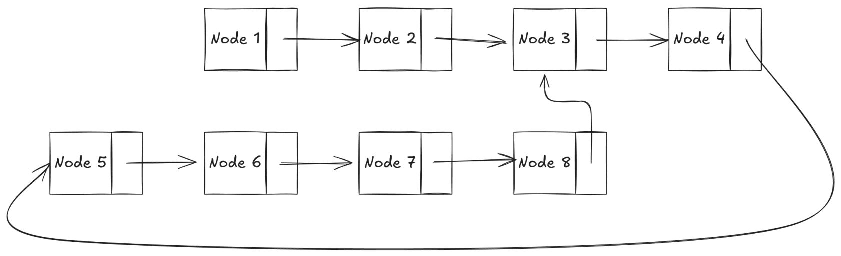
This is an interactive problem.

In computer science, a **linked list** is a linear data structure where elements, or nodes, are connected through links. Each node contains data and a reference (or pointer) to the next node in the sequence. This structure allows the list to grow or shrink dynamically, as nodes are not required to be in contiguous memory locations like an array.

Below is a C++ definition of a linked list:

```
struct Node {
    int value;
    Node* next;  // pointer to the next node
}
struct LinkedList {
    Node* head;  // pointer to the first node of the linked list
}
```

When a node in a linked list points to a previous node in the linked list, it creates a **loop**. Note that it's possible for a node to point to itself, in which case the loop has a length of 1. The following illustration shows a linked list with 8 nodes and a **loop** of length 6.



In this problem, the judge has prepared an interactor, initialized with a head node of a secret linked list of length n which **has a loop**. You do not have any information about this linked list, even its length n. Your goal is to find the length of the loop.

To do this, you can **mark** some nodes, and when visiting a node, you will be informed by the interactor whether a node has been **marked**. More precisely:

- Initially, all nodes are unmarked.
- The interactor has a pointer variable p. At any point in time, p points to some node of the linked list. Initially, p points to the head node.
- You will give instructions to the interactor in at most $3 \cdot n$ rounds. In each round:
  - You tell the interactor whether you want to **permanently mark** the node that p is currently pointing to, or do nothing with this node.
  - The interactor then set p to its next node. In other words, the interactor performs the following assignment: $p \leftarrow p.next$.
  - The interactor then tell you whether the new node p is pointing to has been previously marked.
- After that, you need to determine the length of the loop.

Interaction Protocol

Each test contains multiple test cases. First, your program reads the number of test cases τ $(1 \le \tau \le 10^5)$. For each test case:

- First, your program interacts with the interactor in **at most** $3 \cdot n$ rounds. In each round:
  - Your program prints either 1 if you want to permanently mark the node that p is currently pointing to, or 0 otherwise.
  - Your program then reads a character c which can be either `Y` if the node currently pointed by p (after the interactor the assignment $p \leftarrow p.next$) has been previously marked, or `N` otherwise.
- Then your program prints `-1` ℓ to answer that the length of the loop is ℓ. Your program should continue with the next test case or terminate immediately if this is the last one.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

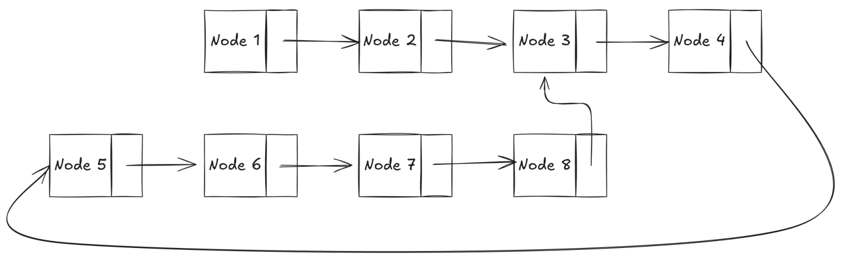
After printing each query do not forget to output the end of line and flush[1] the output.

Sample Interaction

Stdin	Stdout	Comment
1		There is one test case.
		p points to node 1.
	0	You don't mark node 1.
N		p points to node 2. It's not marked.
	0	You don't mark node 2.
N		p points to node 3. It's not marked.
	1	You **mark** node 3.
N		p points to node 4. It's not marked.
	0	You don't mark node 4.
N		p points to node 5. It's not marked.
	0	You don't mark node 5.
N		p points to node 6. It's not marked.
	0	You don't mark node 6.
N		p points to node 7. It's not marked.
	0	You don't mark node 7.
N		p points to node 8. It's not marked.
	0	You don't mark node 8.
Y		p points to node 3. It's **marked**.
	-1 6	You answer that the length of the loop is $\ell = 6$.

The example interaction only shows how your program interacts with the interactor, and does not show how the answer is deduced.

The image below illustrates the linked list used in the sample interaction.



[1]To flush, use: `fflush(stdout)` or `cout.flush()` in C++; `sys.stdout.flush()` in Python; or see the documentation for other languages.