CS1020 Take-home Lab #3

Exercise #1: Alternating List

(http://www.comp.nus.edu.sg/~cs1020/3_ca/labs.html)

Objective

The objective of this exercise is to use the Java API **LinkedList**.

Task Statement

A list is *alternating* if its elements alternate between positive and negative. For example:

Alternating lists	***Non-alternating lists***
[-1, 10, -2, 3, -5]	[-4, 2, -3, -7, 6]
[1, -2, 3, -4]	[4, 3]
[3]	[8, -8, 3, 4, -1]
[ ] (empty list)	[-2, -4, -6]

You are given a linked list (original) and **Q** updates. For each update, you are to check if the updated linked list is *alternating*.

There are 3 valid update operations, as explained below. Note that the indices of the list begin with one, not zero, in this exercise. Hence, the first node is at index 1, not 0.

1. **M <index> <size>**: Move a block of elements of length <size> starting at index <index> to the back of the list. For example, let the current linked list be [1, 3, 5, 4, 2, 6]. The operation "**M 2 3**" moves **[3, 5, 4]** to the end of the linked list. The updated linked list is [1, 2, 6, 3, 5, 4].
2. **R <index> <size>**: Remove a block of elements of length <size> starting at index <index> from the linked list. For example, let the current linked list be [1, 3, 5, 4, 2, 6]. The operation "**R 2 4**" removes **[3, 5, 4, 2]** from the list. After performing the operation, we will have [1, 6].
3. **A <index> <size> <value>**: Add the elements between index **<index>** and **<index + size – 1>** (inclusive) with **<value>**. For example, let the current linked list be [1, -3, -5, 6, 10]. The operation "**A 2 3 4**" adds **[-3, -5, 6]** with value **4**. The updated linked list is [1, **1, -1, 10**, 10].

* ***It is guaranteed that all update operations given are valid.*** For instance, you will not be given the operation "M 2 3" on a list with fewer than 4 nodes. For every operation, you may assume that <size> is positive.

You must use the Java API **LinkedList** in your program.

Input

The first line of the input contains 2 integers **N** ($1 \leq N \leq 100$) and **Q** ($1 \leq Q \leq 100$), where **N** is the size of the original linked list and **Q** the number of updates. The next line contains **N** integers, denoting the elements in the original linked list. The next **Q** lines are the update operations.

Note that short symbols such as **N** and **Q** are used above for convenience. In your program, you are expected to give them descriptive variable names.

Output

Print "**YES**" if the updated linked list is alternating, otherwise print "**NO**".

Sample Input #1

```
4 4
1 -2 3 -4
M 1 3
A 1 1 14
R 2 2
A 2 1 -11
```

Sample Output #1

```
YES
NO
NO
YES
```

Explanation:

Update 1: [1, -2, 3, -4] → [-4, 1, -2, 3]. Updated list is alternating.

Update 2: [-4, 1, -2, 3] → [10, 1, -2, 3]. Updated list is NOT alternating.

Update 3: [10, 1, -2, 3] → [10, 3]. Updated list is NOT alternating.

Update 4: [10, 3] → [10, -8]. Updated list is alternating.