Doubly Linked List Editor

Problem Description

You are given an integer sequence of length N, treated as a doubly linked list. The initial nodes are numbered $1..N$ in the given order and connected head $\rightarrow \cdots \rightarrow$ tail. Then Q operations follow. Maintain the list accordingly and, after all operations, print the sequence from head to tail and from tail to head.

All operations that reference a node id k are guaranteed to be valid at that time (i.e., k exists and has not been deleted). Every newly inserted node receives a fresh id starting from $N{+}1$ and increasing by insertion order. Deleted ids are not reused.

Supported operations (one per line):

- `H x` —Insert a new node with value x at the head.
- `T x` —Insert a new node with value x at the tail.
- `A k x` —Insert a new node with value x after node k.
- `B k x` —Insert a new node with value x before node k.
- `D k` —Delete node k.
- `MH k` —Move node k to the head.
- `MT k` —Move node k to the tail.

Intended approach: maintain a doubly linked list (often implemented via arrays `L[]`/`R[]`/`val[]`) to achieve $O(1)$ amortized time per operation. Do not shift whole arrays.

Input Format

- Line 1: two integers N, Q.
- Line 2: N integers $a_1, a_2, \ldots, a_N$.
- Next Q lines: each line is one operation in the format above.

Notes:

- The initial nodes are $1..N$ with values $a_1..a_N$ in order.
- New node ids start from $N{+}1$ and increase by insertion order.
- All referenced ids are valid at the time of the operation.

Output Format

- Line 1: the values from head to tail (space-separated), or `EMPTY` if the list is empty.
- Line 2: the values from tail to head (space-separated), or `EMPTY` if the list is empty.

Constraints

- $1 \le N, Q \le 2 \times 10^5$.
- $-10^9 \le a_i, x \le 10^9$.
- Time Limit: 1 second; Memory Limit: 256 MB.
- Target complexity: overall $O(N+Q)$; $O(1)$ amortized per operation.

Example Test Case

Sample Input 1

```
3 5
10 20 30
B 2 15
A 3 25
MH 5
D 1
T -7
```

Sample Output 1

```
25 15 20 30 -7
-7 30 20 15 25
```

Explanation 1

Initial list: $[10(\text{id}=1),\ 20(\text{id}=2),\ 30(\text{id}=3)]$

- `B 2 15`: insert 15 before id$=2$ (new id$=4$) $\Rightarrow 10, 15, 20, 30$.
- `A 3 25`: insert 25 after id$=3$ (new id$=5$) $\Rightarrow 10, 15, 20, 30, 25$.
- `MH 5`: move id$=5$ to head $\Rightarrow 25, 10, 15, 20, 30$.
- `D 1`: delete id$=1$ (10) $\Rightarrow 25, 15, 20, 30$.
- `T -7`: insert -7 at tail $\Rightarrow 25, 15, 20, 30, -7$.

Sample Input 2

```
1 4
0
H 5
B 1 -1
D 2
MT 1
```

Sample Output 2

```
-1 0
0 -1
```

Explanation 2

Initial: $[0(\text{id} = 1)]$

- `H 5`: head-insert 5 (id$= 2$) $\Rightarrow 5, 0$.
- `B 1 -1`: insert -1 before id$= 1$ (id$= 3$) $\Rightarrow 5, -1, 0$.
- `D 2`: delete id$= 2$ (5) $\Rightarrow -1, 0$.
- `MT 1`: move id$= 1$ (0) to tail $\Rightarrow -1, 0$ (already at tail).

Sample Input 3

```
2 3
7 7
D 1
D 2
H 100
```

Sample Output 3

```
100
100
```

Explanation 3

Delete both initial nodes $(7, 7)$ so the list becomes empty; then `H 100` inserts 100 at head. A single node prints as `100` in both directions.

Sample Input 4

```
2 2
1 2
D 1
D 2
```

Sample Output 4

```
EMPTY
EMPTY
```

Explanation 4

Delete both nodes of the initial list; the list is empty, so both lines are `EMPTY`.