Deque

Problem ID: deque

Write your own implementation of a double ended deque, which is often referred to as deque, specialized to store 32-bit signed integers.

The data structure should support the following operations:

- Default construction - initializes an empty instance of a deque. This is not tested explicitly here!
- Assignment and copy construction - must properly copy the contents of another instance of the data structure. Ensure the two instances do not share memory afterwards!
- Push front - must insert an element to the front of the deque.
- Push back - must insert an element to the back of the deque.
- Pop front - must remove the front element off of the deque.
- Pop front - must remove the back element off of the deque.
- Front - must provide access to the front element of the deque.
- Back - must provide access to the back element of the deque.
- Size - must provide the size of the deque.

You must avoid any memory leaks or other memory errors in your implementation.

Input

The input starts with a line containing an integer q, representing the number of lines that follow. Each line will represent an operation that is run. The lines have the following format: `<id> <operation> [arg]`

The instance ID will be an integer from 1 to $1,000$, representing an instance of the data structure. Each instance should be default initialized at the start as an empty deque.

The operations are the following:

- `a` - construct a copy of deque, takes integer argument for the instance ID of the deque to copy
- `+f` - push a value to the front of the deque, takes integer argument for the value to push
- `+b` - push a value to the back of the deque, takes integer argument for the value to push
- `-f` - pop the front of the deque, no additional arguments
- `-b` - pop the back of the deque, no additional arguments
- `f` - output the front element of the deque, see output section
- `b` - output the back element of the deque, see output section
- `s` - output size of deque, see output section

You may assume requested operations will not cause an error in a correctly implemented data structure. For example, a pop operation on an empty deque will not occur in the input.

Output

For each front operation, output a line with the front element of the deque.
For each back operation, output a line with the back element of the deque.
For each size operation, output a line with the size of the deque.

Sample Input 1

```
7
1 +b 1
1 f
1 +f 2
1 f
1 s
1 -b
1 s
```

Sample Output 1

```
1
2
2
1
```

Sample Input 2

```
11
1 +f -9
1 +b 6
1 +f 42
2 a 1
2 +b -5
1 s
2 s
1 f
1 b
2 f
2 b
```

Sample Output 2

```
3
4
42
6
42
-5
```