Teque

Problem ID: teque
CPU Time limit: 2 secon
Memory limit: 1024 MB

Author: Tan Jun An
Source: NUS Competitive Programming

You have probably heard about the *deque* (double-ended queue) data structure, which allows for efficient pushing and popping of elements from both the front and back of the queue. Depending on the implementation, it also allows for efficient random access to any index element of the queue as well. Now, we want you to bring this data structure up to the next level, the *teque* (triple-ended queue)!

The *teque* supports the following four operations:

1. **push_back x**: insert the element x into the back of the *teque*.
2. **push_front x**: insert the element x into the front of the *teque*.
3. **push_middle x**: insert the element x into the middle of the *teque*. The inserted element x now becomes the new middle element of the *teque*. If k is the size of the teque before the insertion, the insertion index for x is $(k+1)/2$ (using 0-based indexing).
4. **get i**: prints out the i^{th} index element (0-based) of the *teque*.

Input

The first line contains an integer N ($1 \leq N \leq 10^6$) denoting the number of operations for the *teque*. Each of the next N lines contains a string S, denoting one of the above commands, followed by one integer x. If S is a **push_back**, **push_front**, or **push_middle** command, x ($1 \leq x \leq 10^9$), else for a **get** command, i ($0 \leq i \leq (\text{size of teque}) - 1$). We guarantee that the *teque* is not empty when any **get** command is given.

Output

For each **get i** command, print the value inside the i^{th} index element of the *teque* in a new line.

Warning

The I/O files are large. Please use fast I/O methods.

Sample Input 1

```
9
push_back 9
push_front 3
push_middle 5
get 0
get 1
get 2
push_middle 1
get 1
get 2
```

Sample Output 1

```
3
5
9
5
1
```

Help