Magical List

This question is graded for 0.5%!

Statement

To prepare for the algorithm interview for Google 2021 STEP internship, Uncle Grandpa is solving problems on Leet Code. He came across the following problem:

You are to implement a special list. Initially the list is empty. Let's the length of the list be n, the front element to be l_1 and the back element to be l_n. It should support the following 6 types of operations:

- $pfront\ x$: insert x to the front of the list
- $pback\ x$: insert x to the back of the list
- $dfront$: print the front element of the list and delete it
- $dback$: print the last element of the list and delete it
- cal: print out the result of $\sum_{i=1}^{n}(-1)^{i+1} \times l_i$
- $get\ i$: print out l_i

Do you think our Grandpa is able to solve this problem? No way he can lol. But you can right? Let's solve it!

Constraints

The following constraints apply for all subtasks.

- $1 \leq q \leq 2 \cdot 10^5$
- In each $pfront$ or $pback$ operation, $-10^9 \leq x \leq 10^9$
- In each get operation, $1 \leq i \leq len(list)$
- The list will not be empty when the $dfront$ or the $dback$ operation is applied.

Further subtask-specific constraints will be listed below, under each subtask.

Input

The first line contains a single integer q -– the number of operations
In the next q lines, each will contain one query of the above 6 types.

Output

Print a single integer – the answer to the problem.

Examples

Sample Input	Expected Output

```
8
pfront 5
pback 10
pback 12
cal
pfront 8
dback
cal
get 2
```
	```
	7
	12
	13
	5
	```

We have the state of the list after each operation

After the 1st operation: [5]

After the 2nd operation: [5,10]

After the 3rd operation: [5,10,12]

After the 4th operation: [5,10,12] $\rightarrow$ print $5 - 10 + 12 = 7$

After the 5th operation: [8,5,10,12]

After the 6th operation: [8,5,10] $\rightarrow$ print 12

After the 7th operation: [8,5,10] $\rightarrow$ print $8 - 5 + 10 = 13$

After the 8th operation: [8,5,10] $\rightarrow$ print 5

Subtasks

You may choose to solve any one of the following subtasks.
This accounts for 70% of your score for this problem.

- **Subtask 1 (80%)**:

  To obtain full credit for this subtask, your solution should run in $O(nq)$ time, where n is the maximum length of the list at any given moment.

- **Subtask 2 (100%)**:

  To obtain full credit for this subtask, the complexity for each query of the first 5 types should be $O(1)$, and the complexity for each query of type 6 can be up to $O(n)$.

- **Subtask 3 (110%)**:
  To obtain full credit for this subtask, the complexity of your entire program should be $O(q)$.

Marking Scheme

- Correctness and Efficiency (70%)
  - Refer to the Subtasks section.
- Programming Style (30%)
  - Purpose of methods and statements (5%)
  - Pre- and post- conditions (5%)
  - Modularity (5%)
  - Scoping (5%)
  - Meaningful Identifiers (4%)
  - Variable Naming (1%)
  - Identation (5%)

Notes

1. A skeleton file has been given to help you. You should not create a new file or rename the file provided. You should develop your program using this skeleton file.
2. You are free to define your own helper methods and classes (or remove existing ones) if it is suitable but you must put all the new classes, if any, in the same skeleton file provided.

Skeleton File

You are given the skeleton file `List.java`. You should see the following contents when you open the file:

```java
/**
 * Name          :
 * Matric. No    :
*/

import java.util.*;

public class List {
    private void run() {
        // implement your "main" method here
    }
    public static void main(String args[]) {
        List runner = new List();
        runner.run();
    }
}
```