Problem 1670: Design Front Middle Back Queue

Problem Information

Difficulty: Medium
Acceptance Rate: 56.60%
Paid Only: No
Tags: Array, Linked List, Design, Queue, Data Stream

Problem Description

Design a queue that supports `push` and `pop` operations in the front, middle, and back.

Implement the `FrontMiddleBack` class:

* `FrontMiddleBack()` Initializes the queue. * `void pushFront(int val)` Adds `val` to the **front** of the queue. * `void pushMiddle(int val)` Adds `val` to the **middle** of the queue. * `void pushBack(int val)` Adds `val` to the **back** of the queue. * `int popFront()` Removes the **front** element of the queue and returns it. If the queue is empty, return `-1`. * `int popMiddle()` Removes the **middle** element of the queue and returns it. If the queue is empty, return `-1`. * `int popBack()` Removes the **back** element of the queue and returns it. If the queue is empty, return `-1`.

Notice that when there are **two** middle position choices, the operation is performed on the **frontmost** middle position choice. For example:

* Pushing `6` into the middle of `[1, 2, 3, 4, 5]` results in `[1, 2, _6_ , 3, 4, 5]`. * Popping the middle from `[1, 2, _3_ , 4, 5, 6]` returns `3` and results in `[1, 2, 4, 5, 6]`.

Example 1:

Input: ["FrontMiddleBackQueue", "pushFront", "pushBack", "pushMiddle", "pushMiddle", "popFront", "popMiddle", "popMiddle", "popBack", "popFront"] [[], [1], [2], [3], [4], [], [], [], [], []] **Output:** [null, null, null, null, null, 1, 3, 4, 2, -1] **Explanation:** FrontMiddleBackQueue q = new FrontMiddleBackQueue(); q.pushFront(1); // [_1_] q.pushBack(2); // [1, _2_] q.pushMiddle(3); // [1, _3_ , 2] q.pushMiddle(4); // [1, _4_ , 3, 2] q.popFront(); // return 1 -> [4, 3, 2] q.popMiddle(); // return 3 -> [4, 2] q.popMiddle(); // return 4 -> [2] q.popBack(); // return 2

-> [] q.popFront(); // return -1 -> [] (The queue is empty)

Constraints:

* `1 <= val <= 109` * At most `1000` calls will be made to `pushFront`, `pushMiddle`, `pushBack`, `popFront`, `popMiddle`, and `popBack`.

Code Snippets

C++:

```
class FrontMiddleBackQueue {
public:
FrontMiddleBackQueue() {

}

void pushFront(int val) {

}

void pushMiddle(int val) {

}

void pushBack(int val) {

}

int popFront() {

}

int popMiddle() {

}

int popBack() {

}
};
```

```
/**
* Your FrontMiddleBackQueue object will be instantiated and called as such:
* FrontMiddleBackQueue* obj = new FrontMiddleBackQueue();
* obj->pushFront(val);
* obj->pushMiddle(val);
* obj->pushBack(val);
* int param_4 = obj->popFront();
* int param_5 = obj->popMiddle();
* int param_6 = obj->popBack();
*/
```

Java:

```
class FrontMiddleBackQueue {

public FrontMiddleBackQueue() {

}

public void pushFront(int val) {

}

public void pushMiddle(int val) {

}

public void pushBack(int val) {

}

public int popFront() {

}

public int popMiddle() {

}

public int popBack() {
```

```
}
}

/**
* Your FrontMiddleBackQueue object will be instantiated and called as such:
* FrontMiddleBackQueue obj = new FrontMiddleBackQueue();
* obj.pushFront(val);
* obj.pushMiddle(val);
* obj.pushBack(val);
* int param_4 = obj.popFront();
* int param_5 = obj.popMiddle();
* int param_6 = obj.popBack();
*/
```

Python3:

```
class FrontMiddleBackQueue:

def __init__(self):


def pushFront(self, val: int) -> None:


def pushMiddle(self, val: int) -> None:


def pushBack(self, val: int) -> None:


def popFront(self) -> int:


def popMiddle(self) -> int:


def popBack(self) -> int:



# Your FrontMiddleBackQueue object will be instantiated and called as such:
# obj = FrontMiddleBackQueue()
```

```
# obj.pushFront(val)
# obj.pushMiddle(val)
# obj.pushBack(val)
# param_4 = obj.popFront()
# param_5 = obj.popMiddle()
# param_6 = obj.popBack()
```