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.