push_front(6)

front_ ●
back_ ●

data_
next_ ●
prev_

data_ 2
next_ ●
prev_ ●

data_ 4
next_ ●
prev_ ●

data_ 3
next_ ●
prev_ ●

data_
next_
prev_ ●

Push-Front(6)

data_ 6
next_ ●
prev_ ●

front_ ●
back_ ●

data_ ☑
next_ ●
prev_ ☑

data_ 2
next_ ●
prev_ ●

data_ 4
next_ ●
prev_ ●

data_ 3
next_ ●
prev_ ●

data_ ●
next_ ☑
prev_ ●

data_ 6
next_ ●
prev_ ●

front_ ●
back_ ●

data_ ●
next_ ●
prev_ ☑

data_ 2
next_ ●
prev_ ●

data_ 4
next_ ●
prev_ ●

data_ 3
next_ ●
prev_ ●

data_ ●
next_ ●
prev_ ●

push_back(6)

Push - back (6)

push_front(6)

push - front(6)

push_back(6)

push-back(6)

pop_front()

front_ ●
back_ ●

data_
next_ ●
prev_

data_
next_
prev_ ●

pop_back()

front_ ●
back_ ●

data_
next_ ●
prev_

data_
next_
prev_ ●

Pop-front()

This is an empty list
so, nothing is to be done

front_ ●
back_ ●

data_ □
next_ ●
Prev_ ☒

data_ ☒
next_ ☒
Prev_ ●

Pop-back()

Empty list
Nothing is to be done

front_ ●
back_ ●

data_ ☒
next_ ●
Prev_ ☒

data_ ☒
next_ ☒
Prev_ ●

pop_front()

Pop_front()

pop_back()

Pop_back()

pop_front()

Pop_front()

pop_back()

pop_back()

insert_after(6,node)

Insert_after(6, node)

insert_after(6,node)

Insert_after(6, node)

insert_after(6,node)

Insert_after (6, node)

insert_after(6,node)

node

As no node is specified, so the new node will be inserted in the beginning of list.

Stack: In the diagrams below list what data members you need to track and what their values are in its initial state and their state after each of the operations are applied to the diagram. If the array needs to be resized, draw the new array with the correct capacity

stack.push(6)

3 is at top of stack

2	3

As it is a stack, so the new element 6 will be pushed at the top of stack.

2	3	6	← Top

```
stack.pop()
stack.pop()
stack.push(6)

initially 5 is at top of stack
```

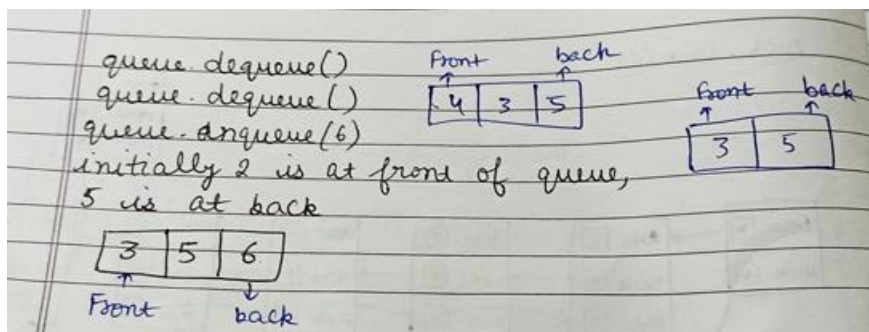| 2	4	3	5

In a stack, the element at the top position is popped out first.

Stack. pop ()
Stack. pop ()
Stack. push (6)
intially 5 is at top of stack

| 2 | 4 | 3 | ← Top

| 2 | 4 | ← Top

| 2 | 4 | 6 | ← Top

Queues: In the diagrams below list what data members you need to track and what their values are in its initial state and their state after each of the operations are applied to the diagram. If the array needs to be resized, draw the new array with the correct capacity

queue.enqueue(6)

2 is at front of queue, 3 is at back

2	3

The new element is added at the back of the list in a queue.

queue. enqueue (6)
2 is at front of queue, 3 is at back

| 2 | 3 | 6 |

```
queue.dequeue()
queue.dequeue()
queue.enqueue(6)

initially 2 is at front of queue,
        5 is at back
```
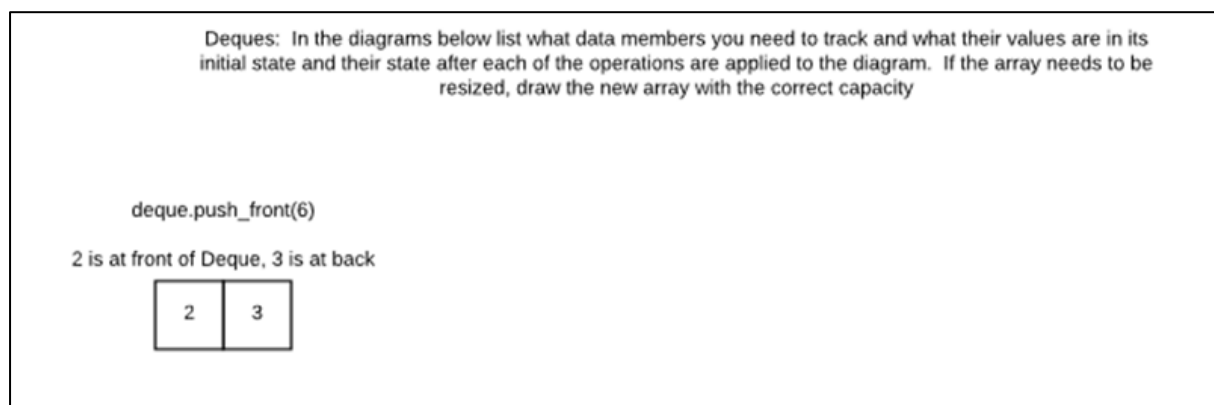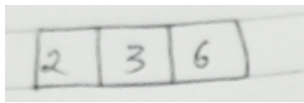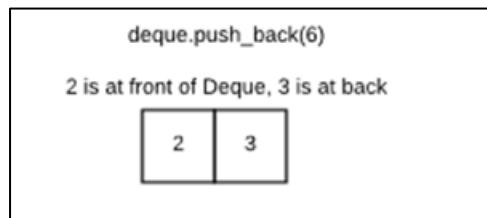
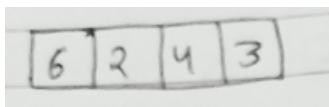| 2 | 4 | 3 | 5 |
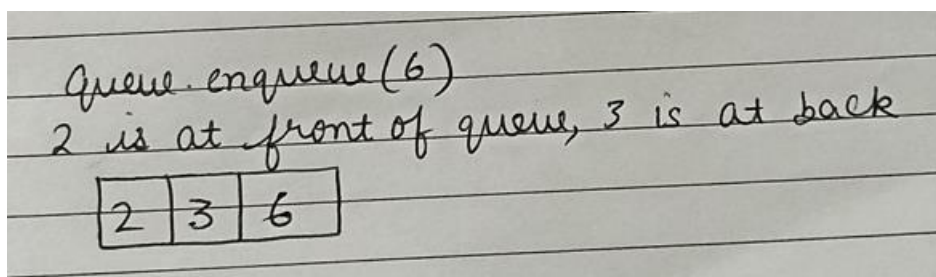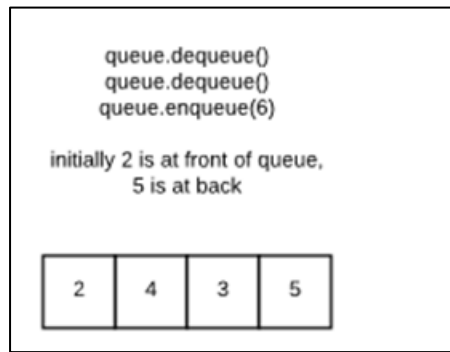
Queue follows the First In First Out (FIFO) structure.

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Deques: In the diagrams below list what data members you need to track and what their values are in its initial state and their state after each of the operations are applied to the diagram. If the array needs to be resized, draw the new array with the correct capacity

```
deque.push_front(6)

2 is at front of Deque, 3 is at back
```

| 2 | 3 |
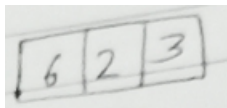
The element is pushed at the front of the list.

deque.push_back(6)

2 is at front of Deque, 3 is at back

2	3

2	3	6

The element is pushed at the back of the list.

deque.pop_back()
deque.push_front(6)

initially 2 is at front of deque, 5 is at back

2	4	3	5

6	2	4	3
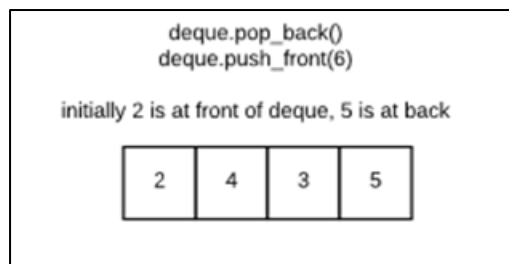
The pop.back() function removes element 5, which is at the back initially, and adds element 6 in the front of the list.

deque.pop_front()
deque.push_back(6)
deque.pop_front()
deque.push_back(7)

initially 2 is at front of deque,
5 is at back

2	4	3	5

4	3	5

4	3	5	6

3	5	6

3	5	6	7
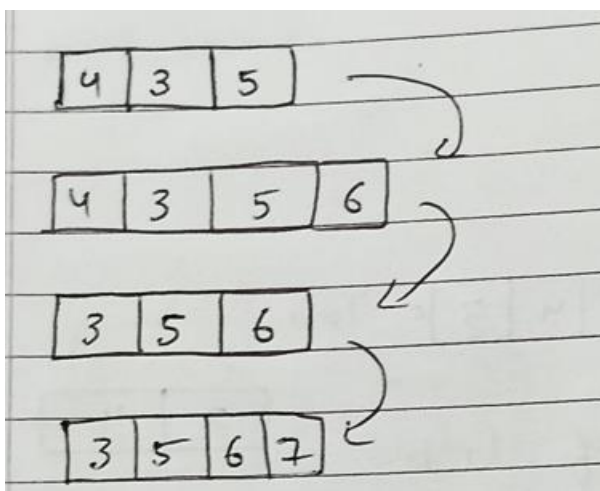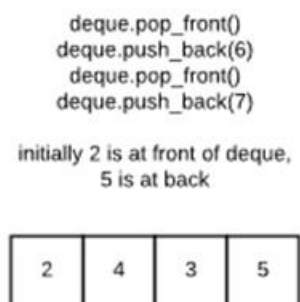
overflow(grid,the_queue) - apply the overflow function to the gride below and show all the grids the function would add to the queue. Number the grid in the order they are added to the queue. Also state the return value. Note that some grids may remain empty

0

-2	1	-3	-3	0
2	0	3	2	0
0	0	-3	0	0
0	0	1	0	0

1

0	-3	-1	-1	-1
-3	0	-4	-3	0
0	0	-3	0	0
0	0	1	0	0

2

-2	0	-3	-1	-1
0	-3	0	-4	0
-1	0	-4	0	0
0	0	1	0	0

3

0	-2	0	-3	-1
-1	-3	-3	0	-1
-1	-1	0	-2	0
0	0	-2	0	0

4

0	-2	-1	0	-2
-1	-3	-3	-1	-1
-1	-1	0	-2	0
0	0	-2	0	0

5

0	-2	-1	-1	0
-1	-3	-3	-1	-2
-1	-1	0	-2	0
0	0	-2	0	0