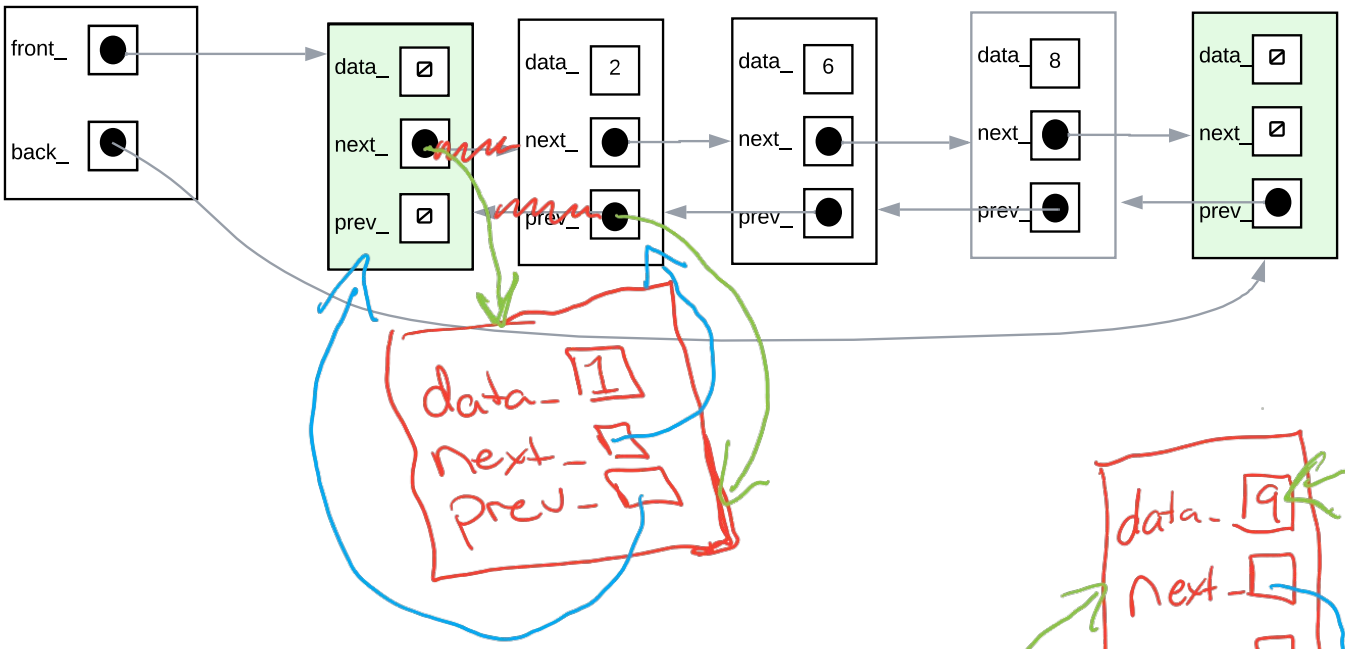
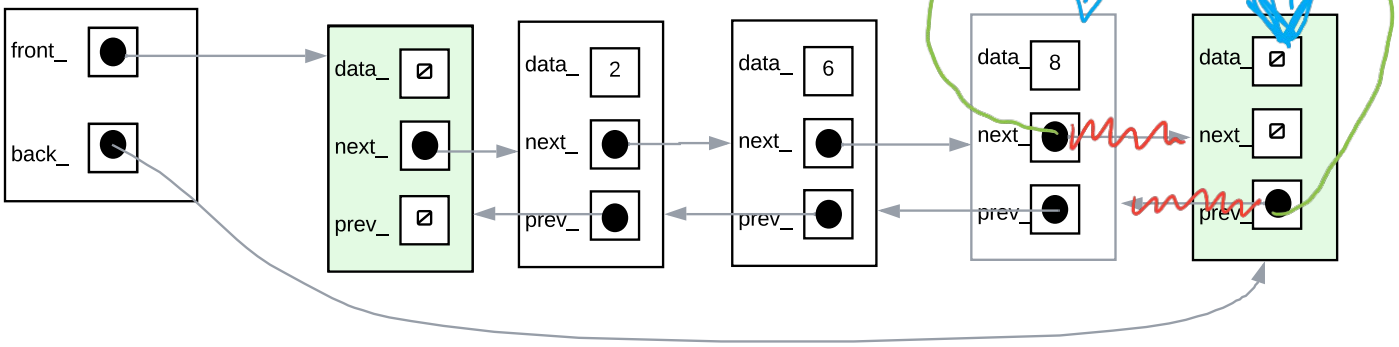
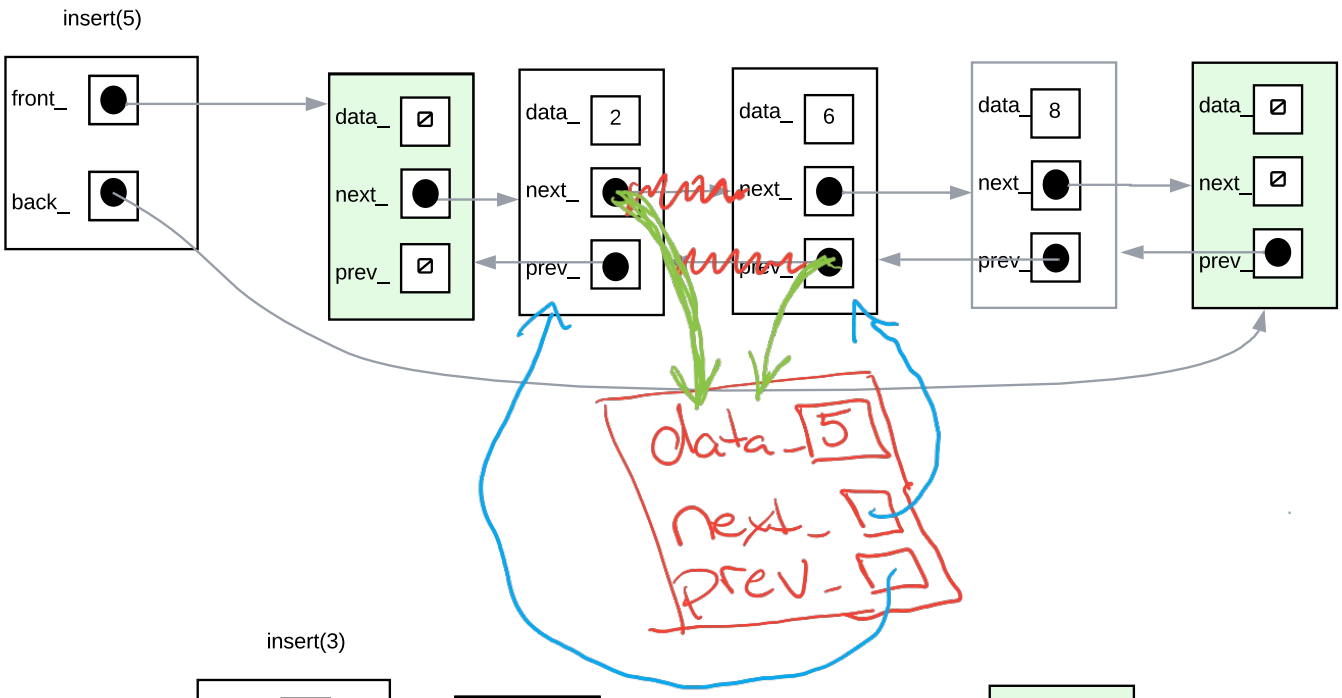
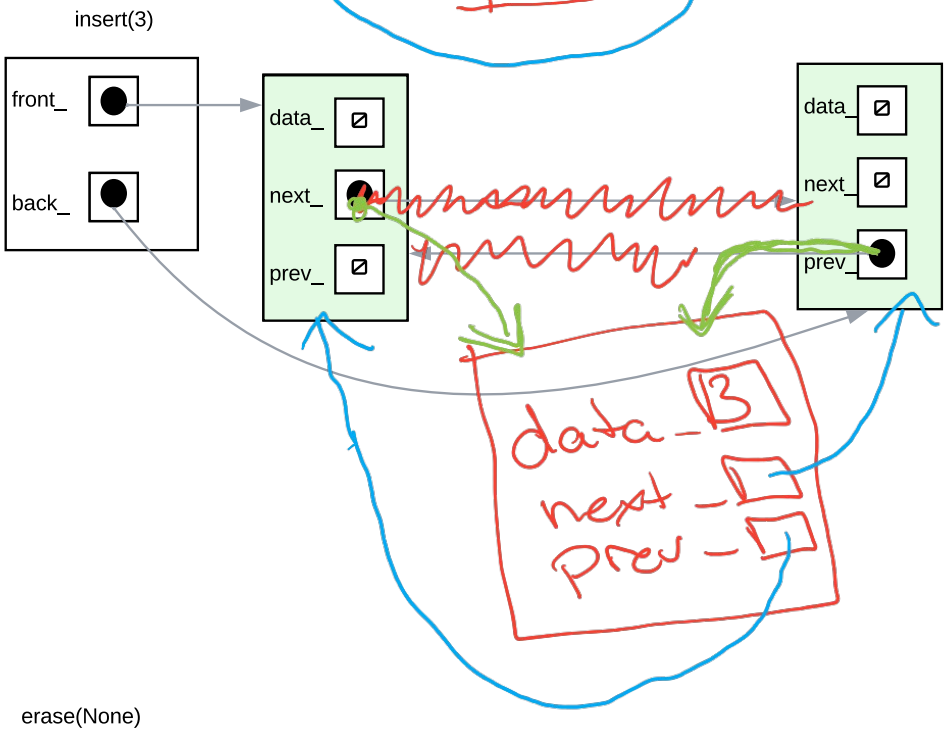
insert(1)

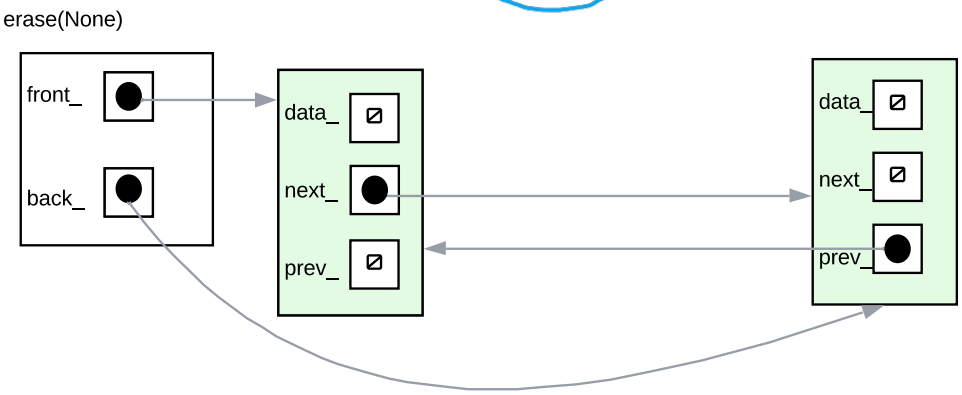


insert(9)

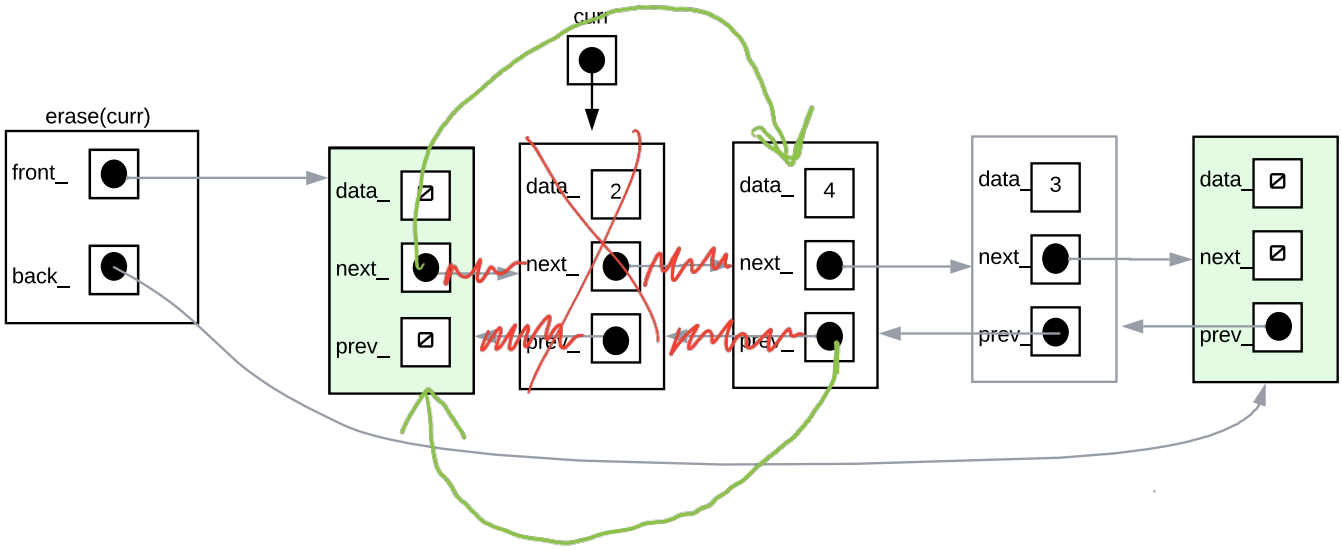


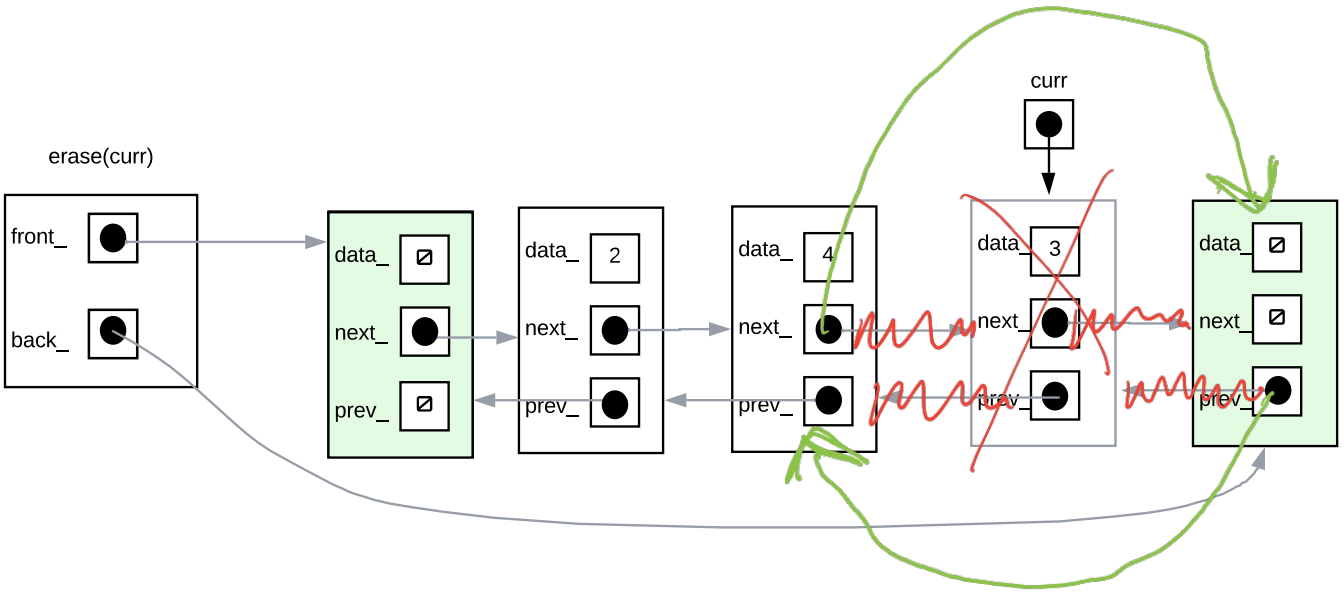


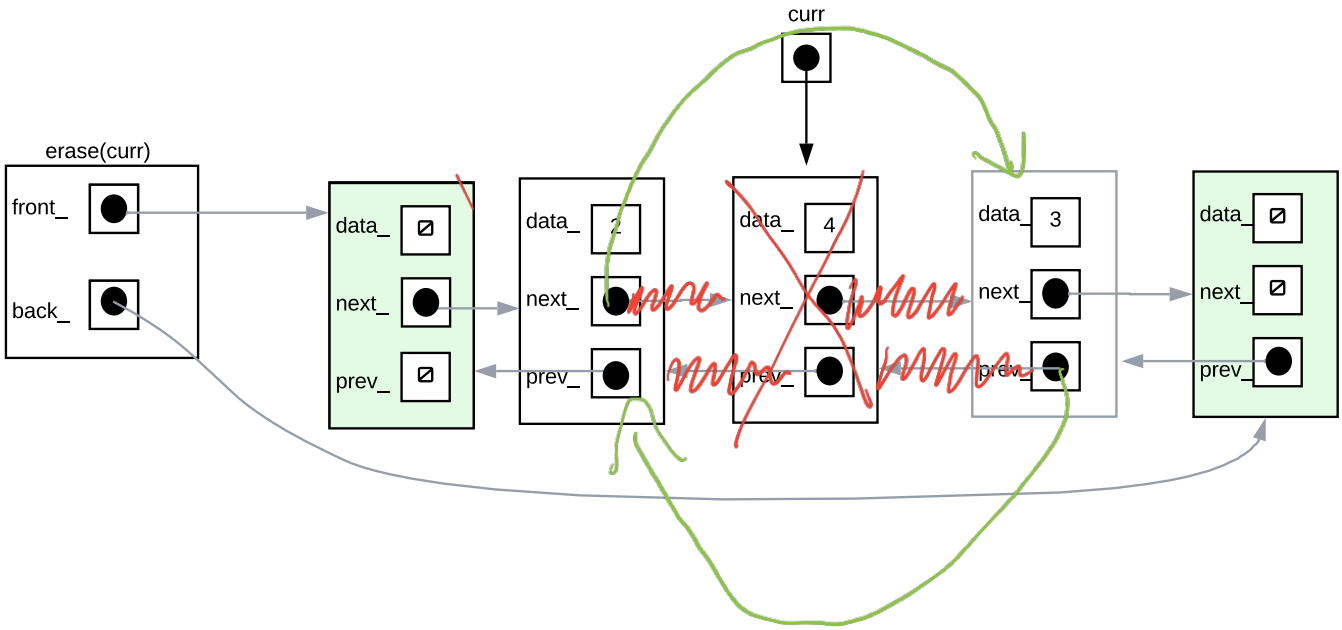


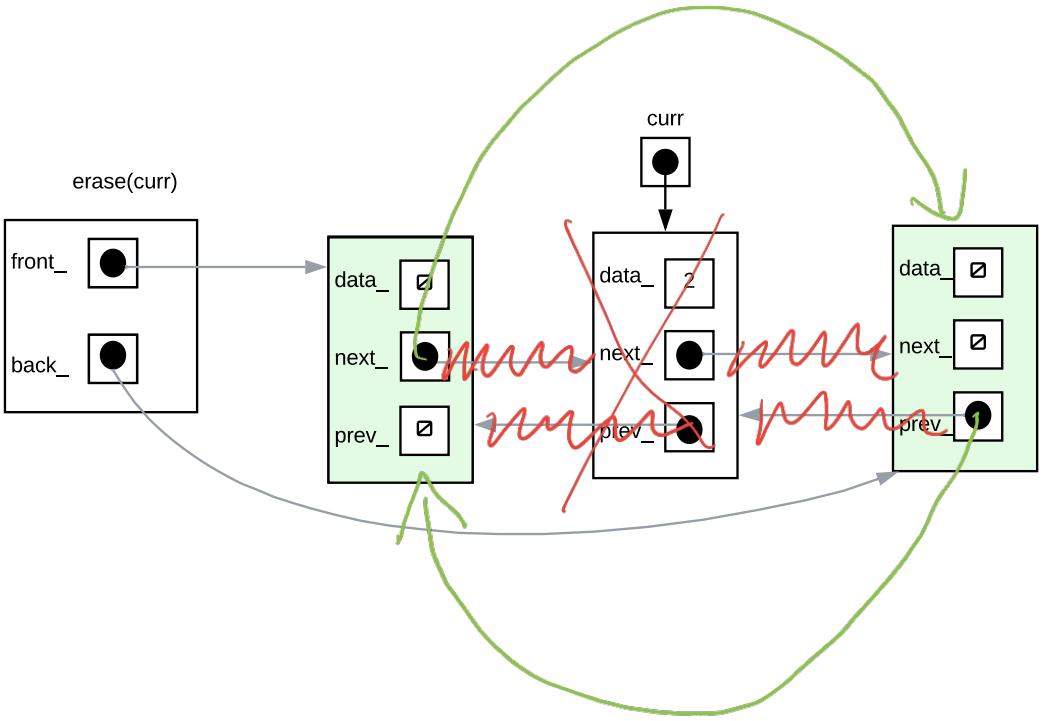


raise ValueError('Cannot erase node referred to by None')


curr
erase(curr)
front_
back_
data_
next_
prev_
data_ 2
next_
prev_
data_ 4
next_
prev_
data 3
next_
prev
data_
next_
prev_


curr
erase(curr)
front_
back_
data_
next_
prev_
data_ 2
next_
prev_
data_ 4
next_
prev_
data_ 3
next_
prev_
data_
next_
prev_

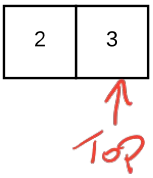

curr
erase(curr)
front_
back_
data_
next_
prev_
2
4
3


curr
erase(curr)
front_
back_
data_
next_
prev_
2

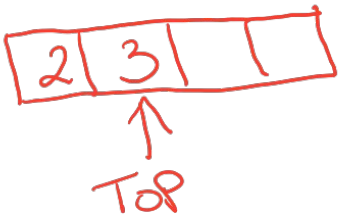
Stack: In the diagrams below list what data members you need to track and what their values are in its initial state and their state after each of the operations are applied to the diagram. If the array needs to be resized, draw the new array with the correct capacity

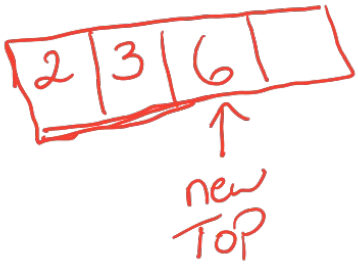
stack.push(6)

3 is at top of stack



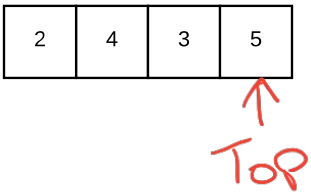
double array size



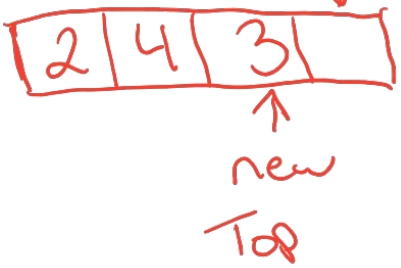


stack.pop()
stack.pop()
stack.push(6)

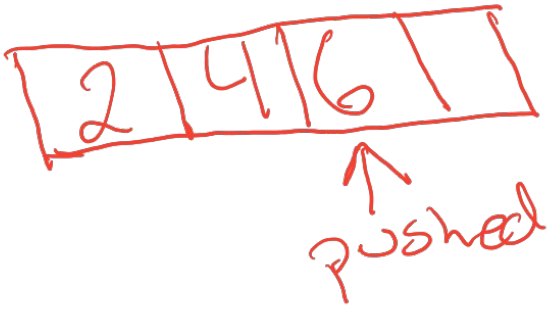
initially 5 is at top of stack



5 was popped



3 was popped

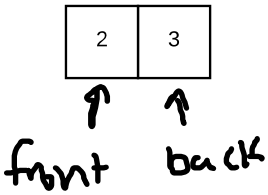


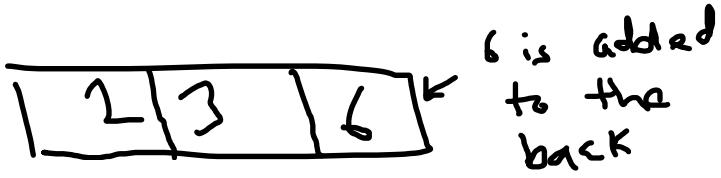
6 is now new Top

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)

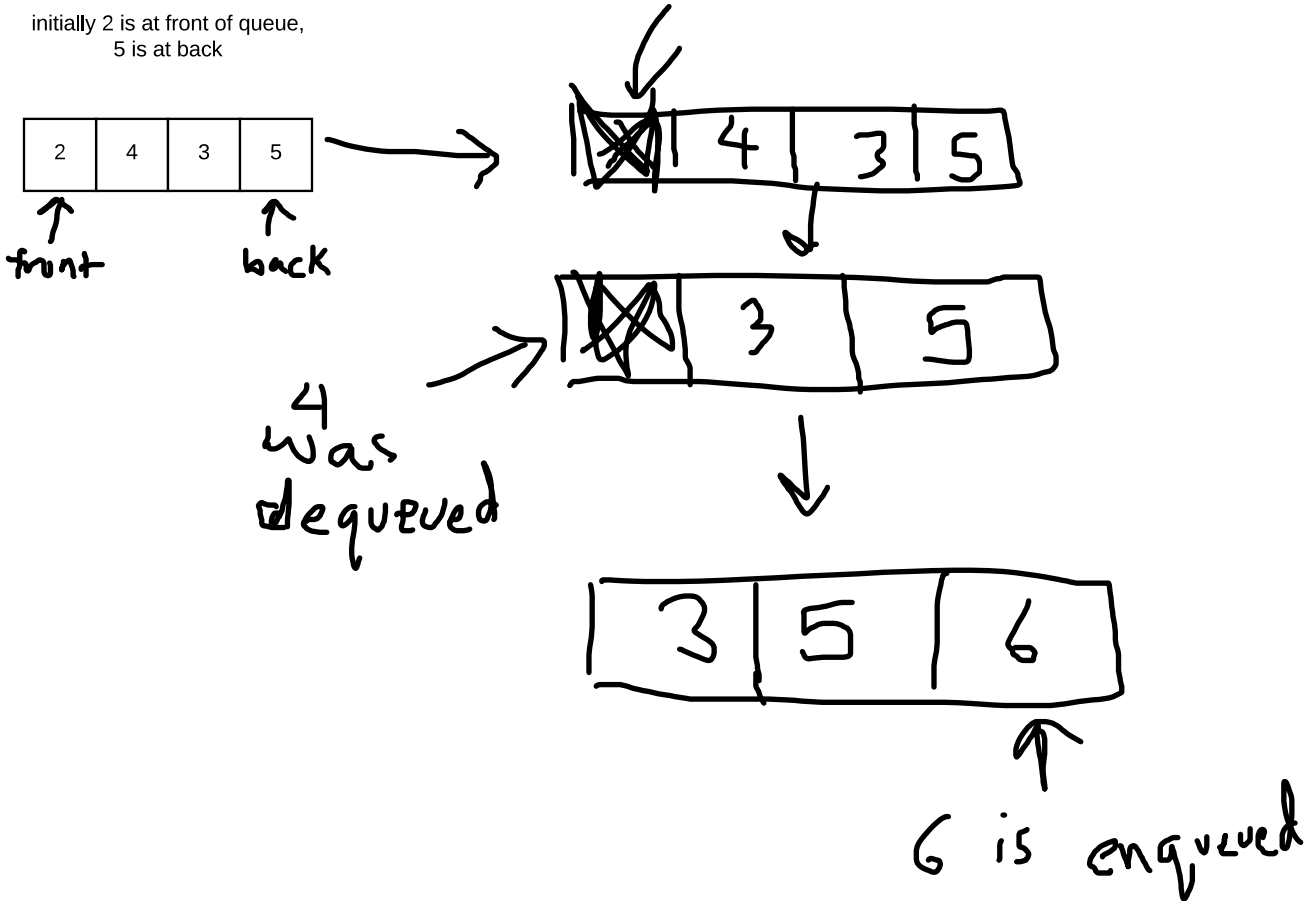
2 is at front of queue, 3 is at back





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

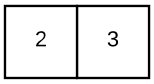
initially 2 is at front of queue, 5 is at back

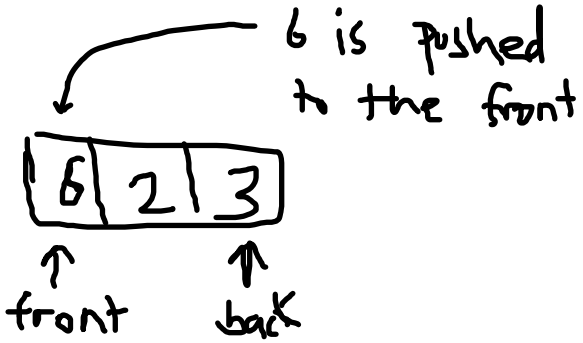


Deques: In the diagrams below list what data members you need to track and what their values are in its initial state and their state after each of the operations are applied to the diagram. If the array needs to be resized, draw the new array with the correct capacity

deque.push_front(6)

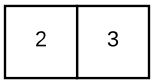
2 is at front of Deque, 3 is at back

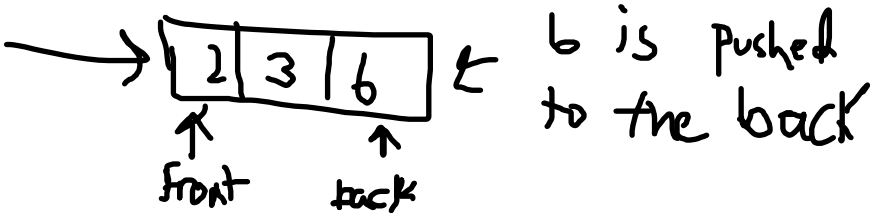




deque.push_back(6)

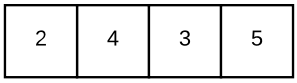
2 is at front of Deque, 3 is at back

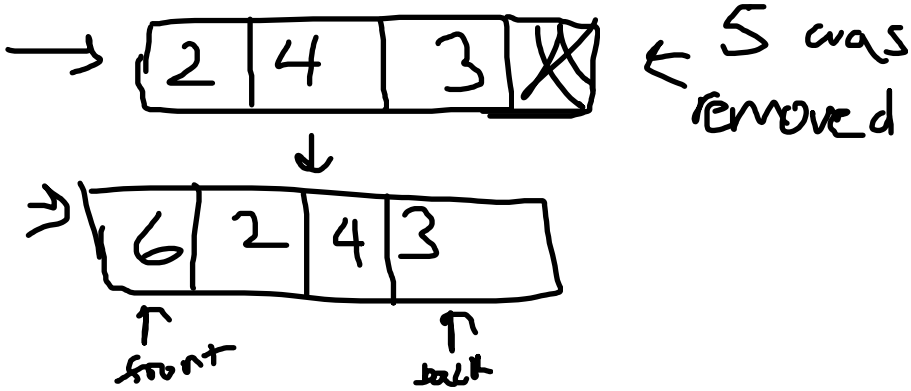




deque.pop_back()
deque.push_front(6)

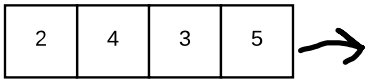
initially 2 is at front of deque, 5 is at back





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



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

[(0,0),(0,2),(0,3)]

4

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

[(0,4)]

1

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

[(0,1)(1,0),(1,2)]

5

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

None

2

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

[(0,0),(0,2),(1,3),(2,2)]

3

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

[(0,3)]
