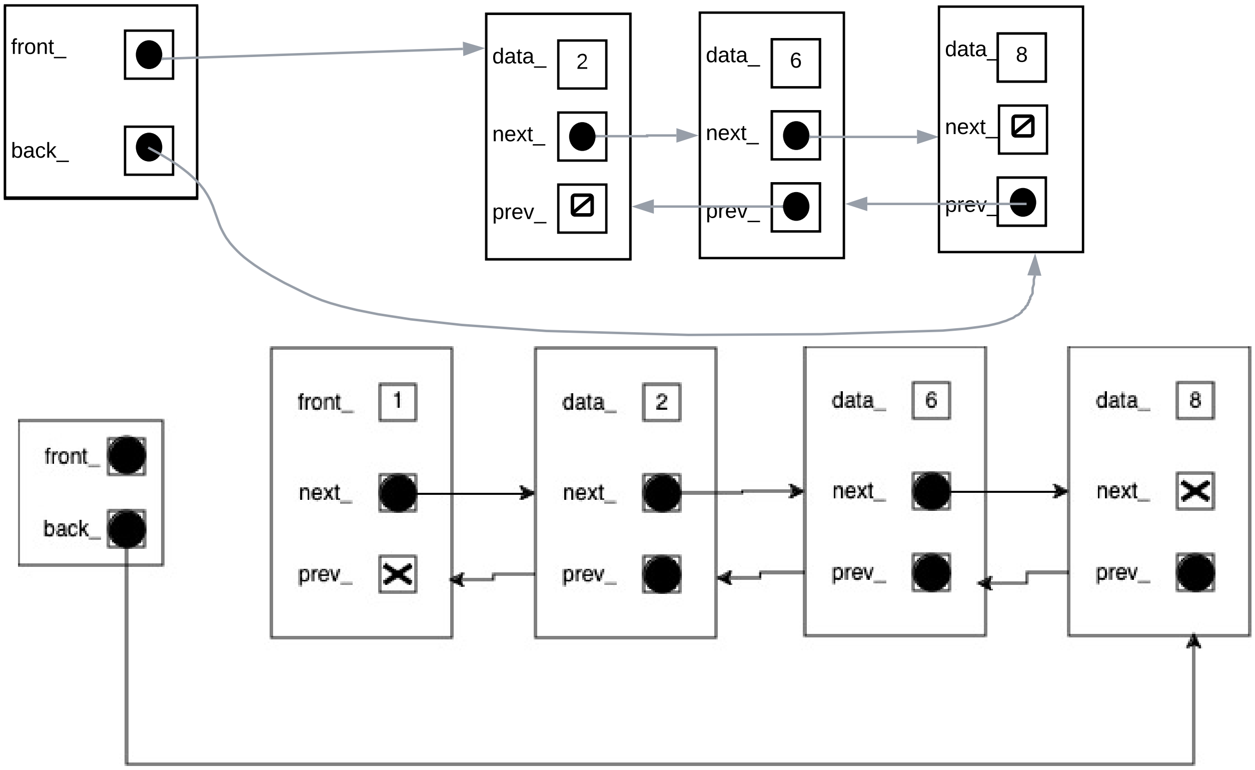
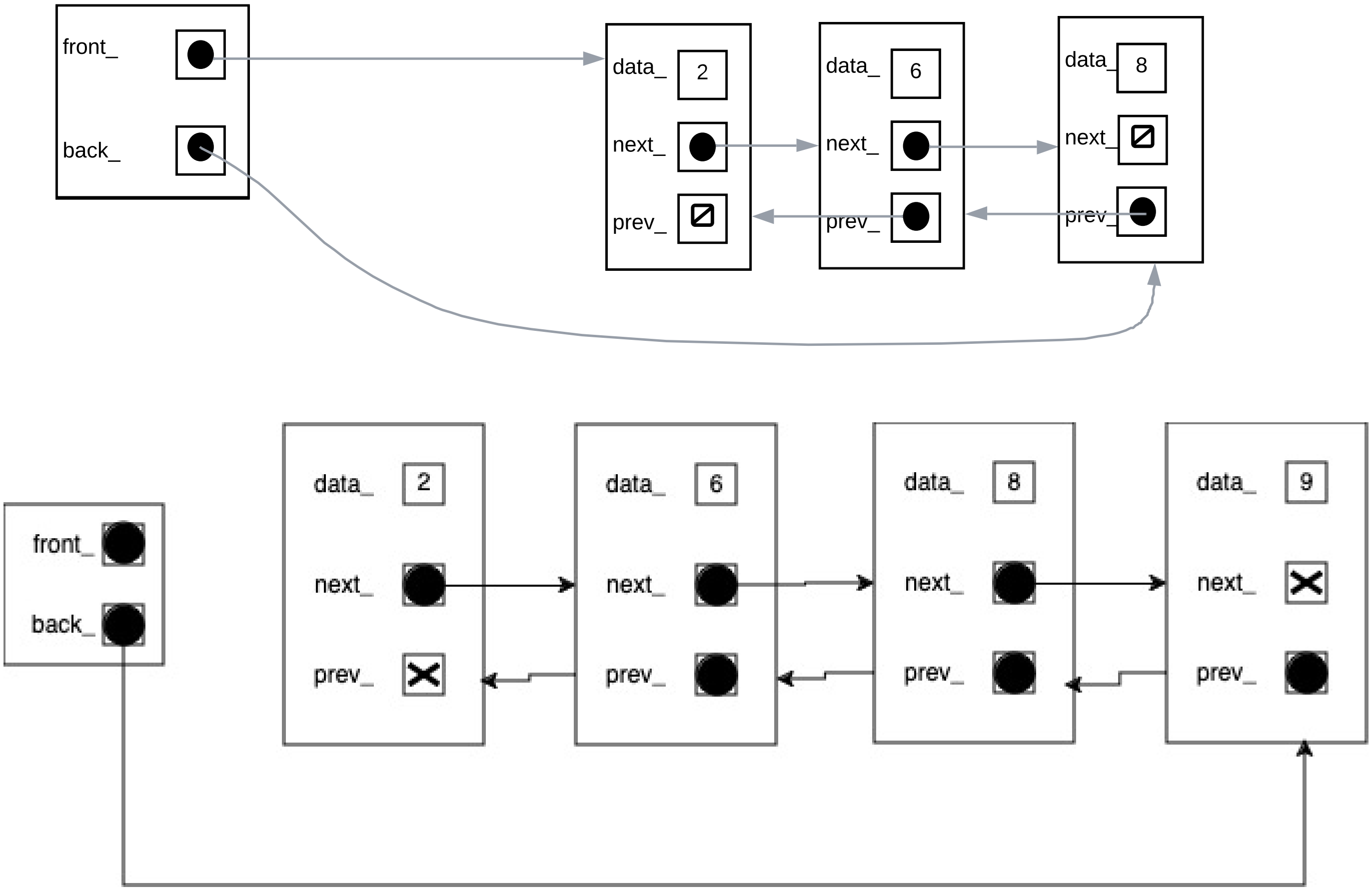
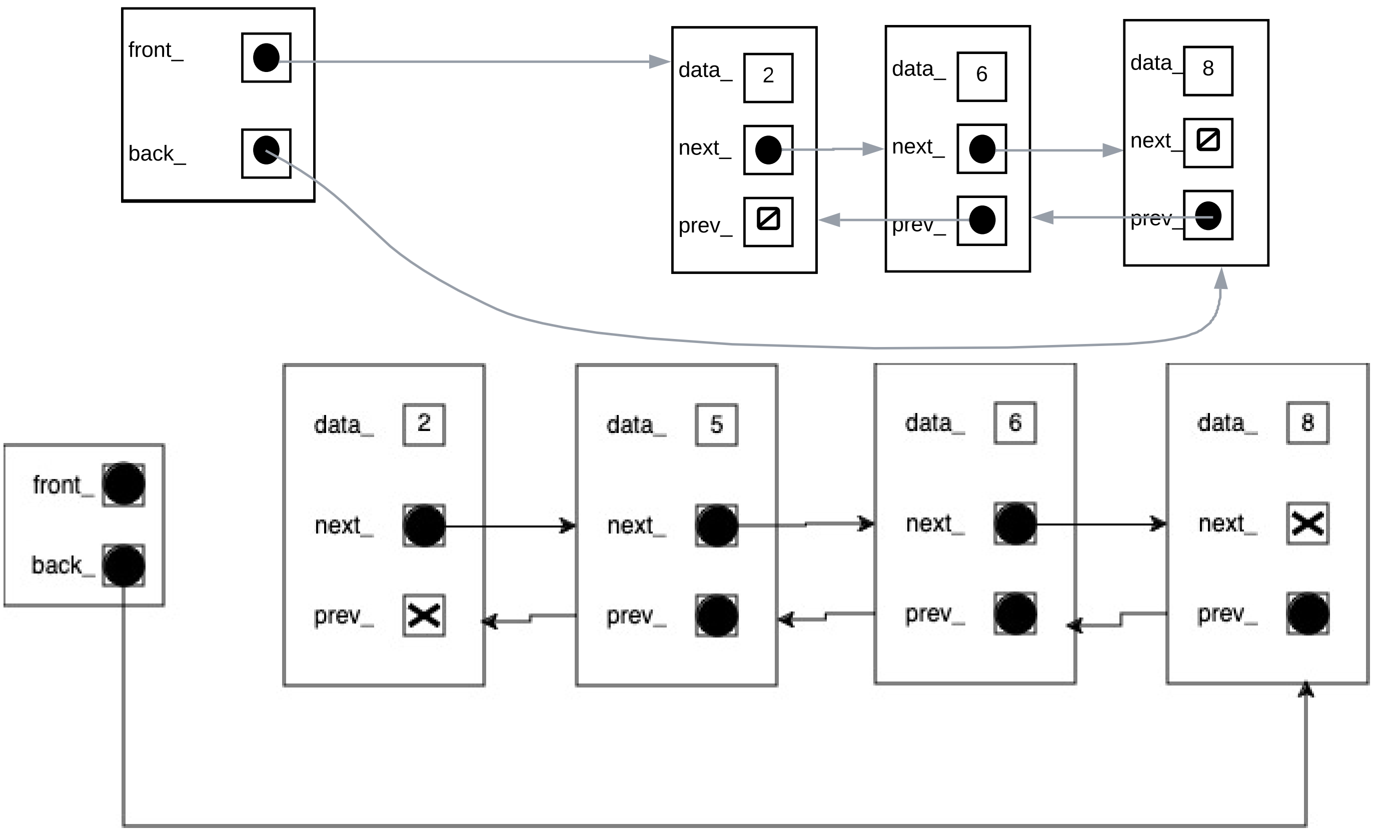
insert(1)



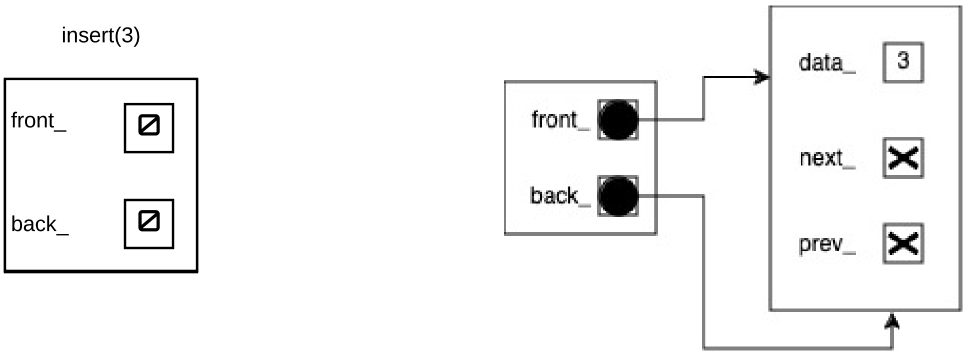
insert(9)



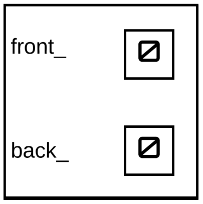
insert(5)



insert(3)

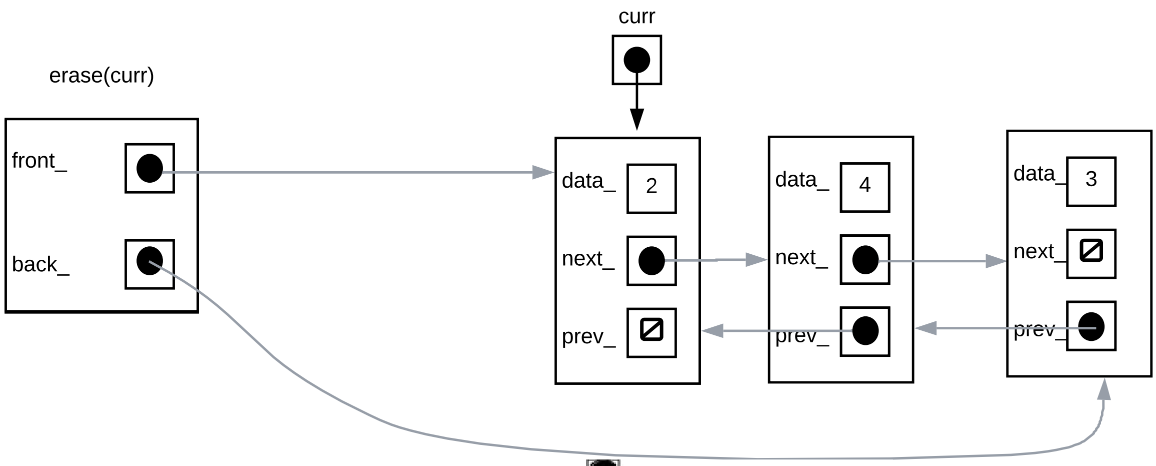


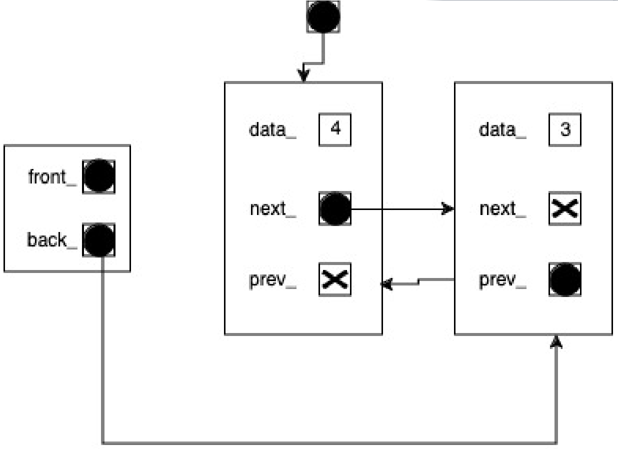
erase(None)



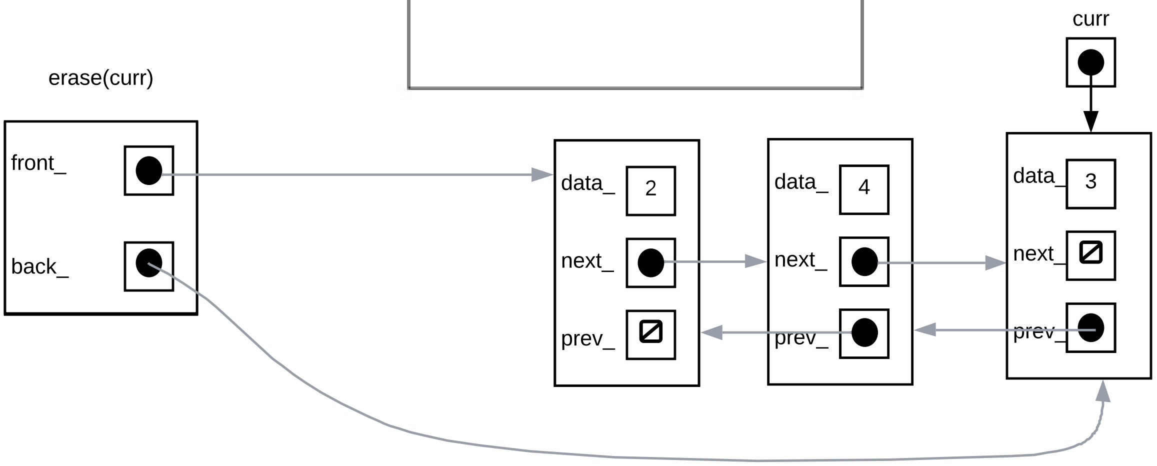
False

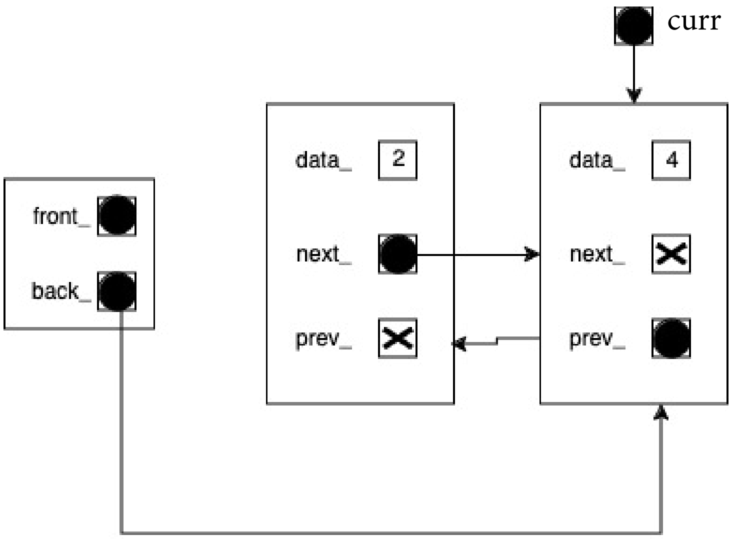
erase(curr)



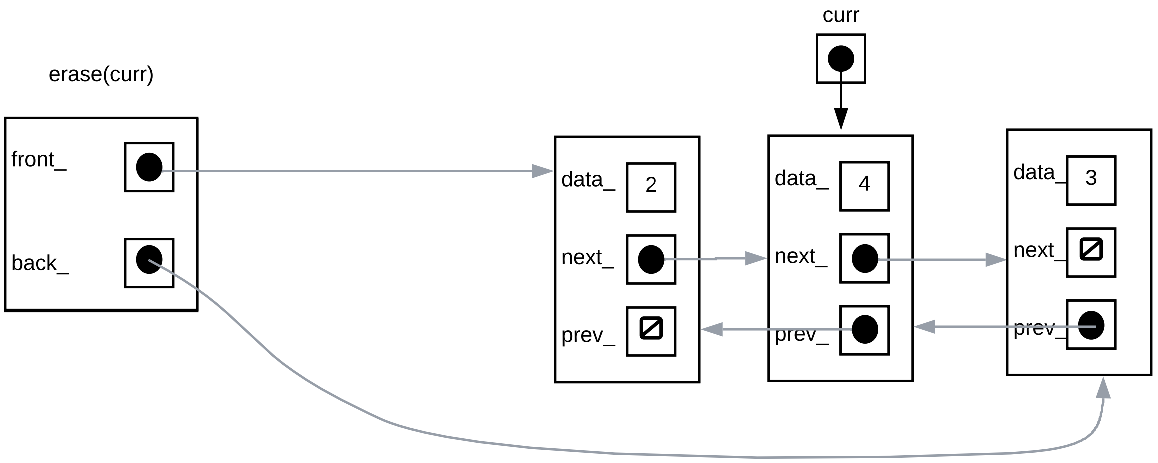


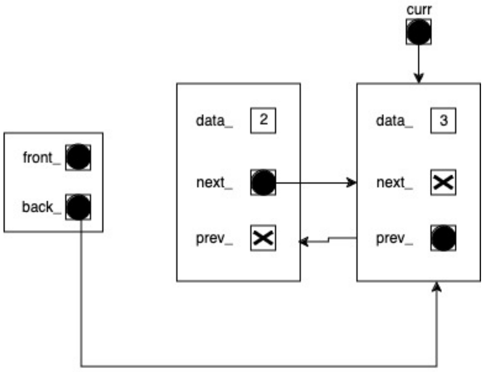
erase(curr)



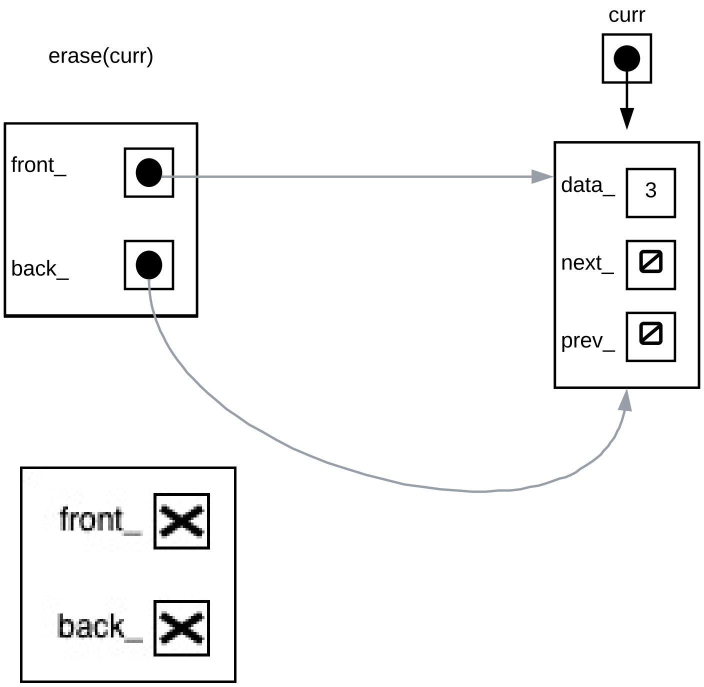


erase(curr)





erase(curr)



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

-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	-2	0	0

2

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

3

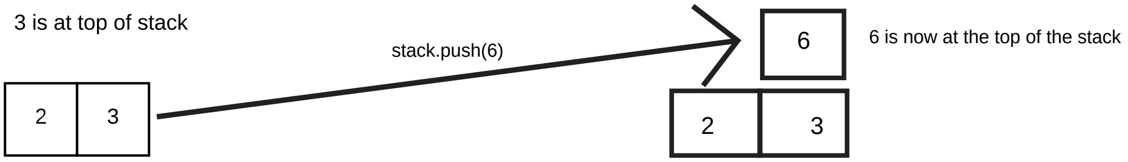
0	-2	0	-3	-1
-1	-3	-3	0	-1
-1	-1	0	-2	0
0	0	-2	0	0

The return value is 3

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

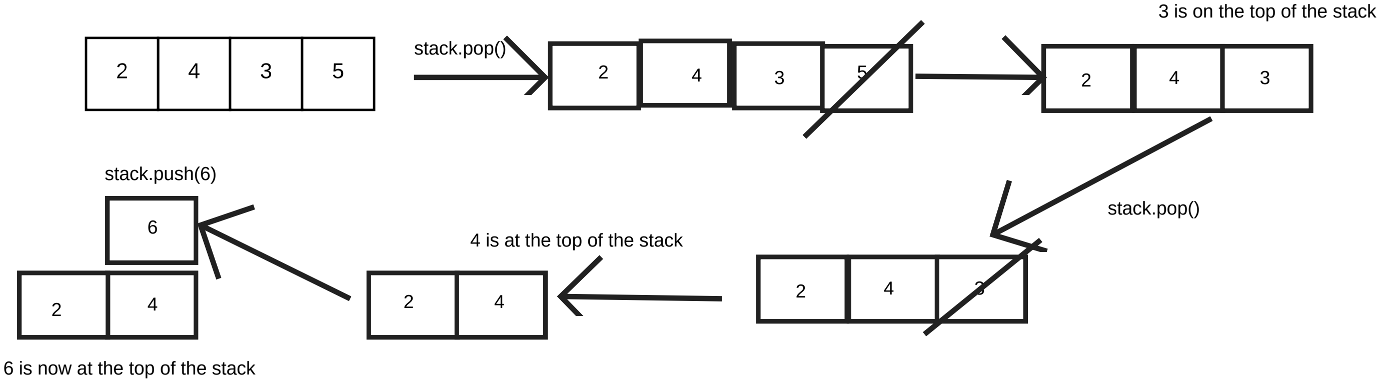
stack.push(6)

3 is at top of stack



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

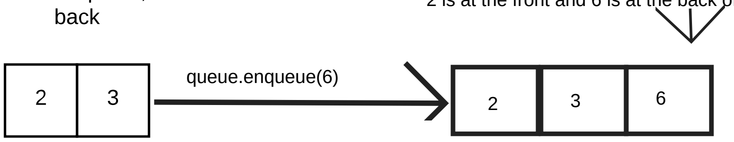
initially 5 is at top of stack


3 is on the top of the stack
2
4
3
5
stack.pop()
2
4
3
5
2
4
3
stack.push(6)
6
stack.pop()
4 is at the top of the stack
2
4
2
4
2
4
3
6 is now at the top of the stack

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

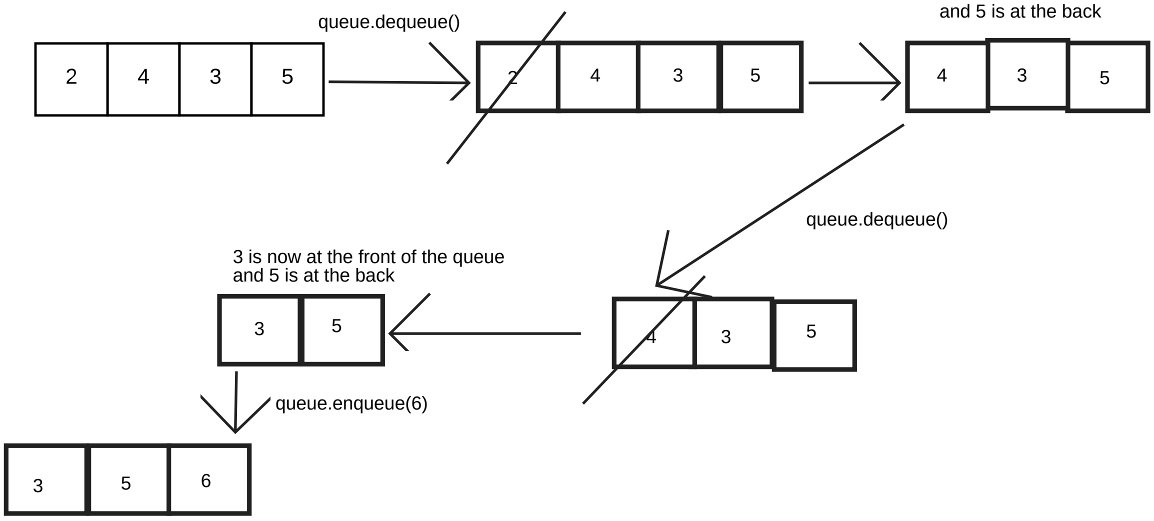


queue.enqueue(6)

2 is at the front and 6 is at the back of the queue

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

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



queue.dequeue()

4 is now at the front of the Queue and 5 is at the back

queue.dequeue()

3 is now at the front of the queue and 5 is at the back

queue.enqueue(6)

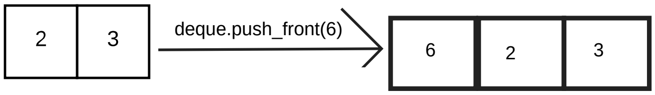
3 is at the front and 6 is at the back

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

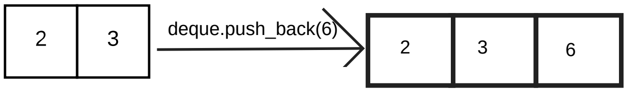
6 is at the front and 3 is at the back



deque.push_back(6)

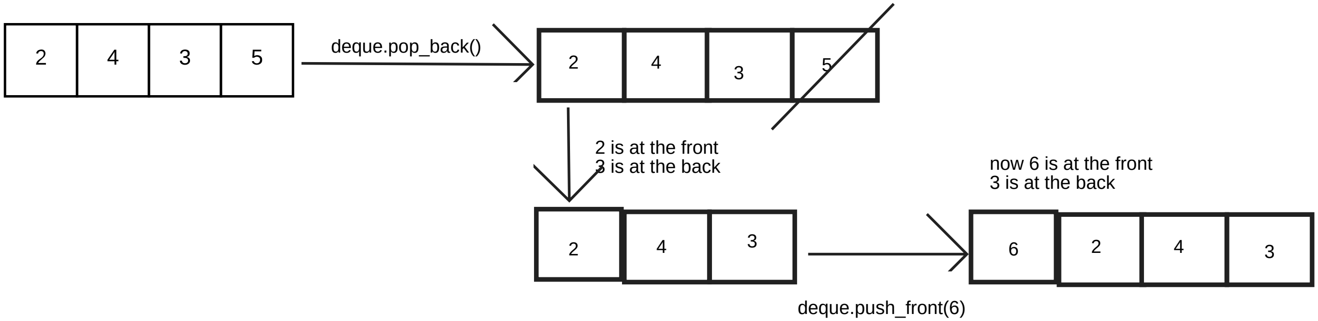
2 is at front of Deque, 3 is at back

2 is at the front and 6 is at the back



deque.pop_back()
deque.push_front(6)

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



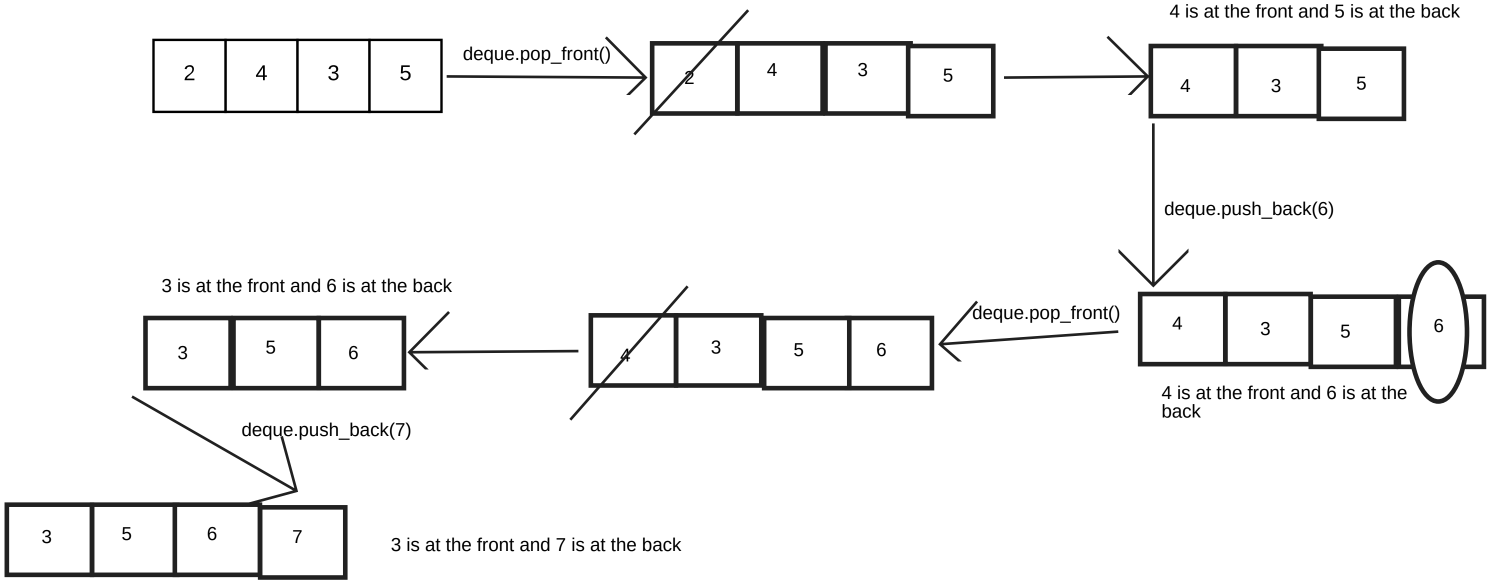
2 is at the front
3 is at the back

now 6 is at the front
3 is at the 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

4 is at the front and 5 is at the back



4 is at the front and 6 is at the back

3 is at the front and 6 is at the back

3 is at the front and 7 is at the back