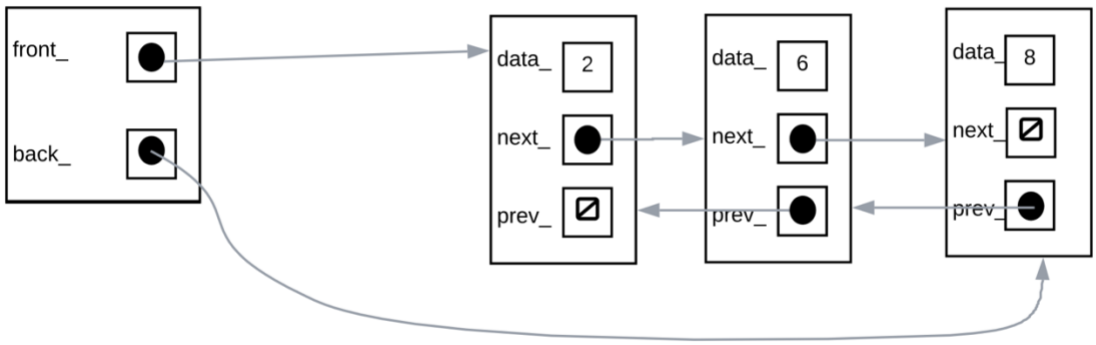
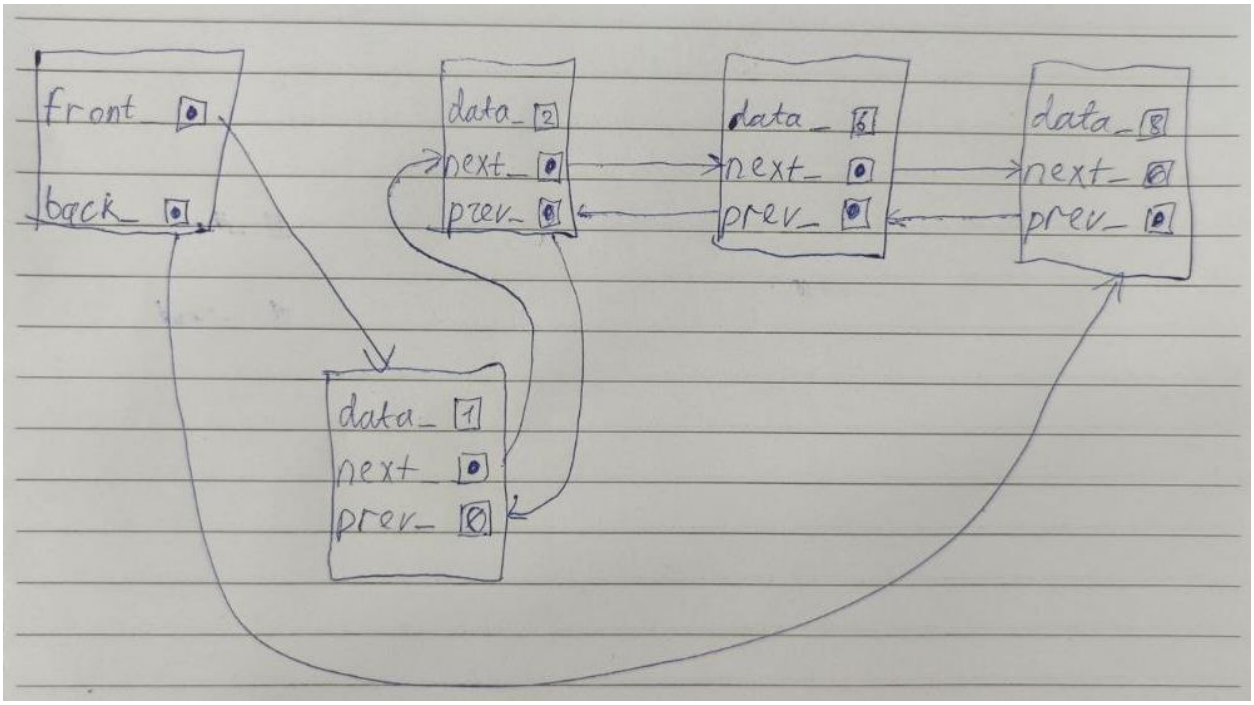
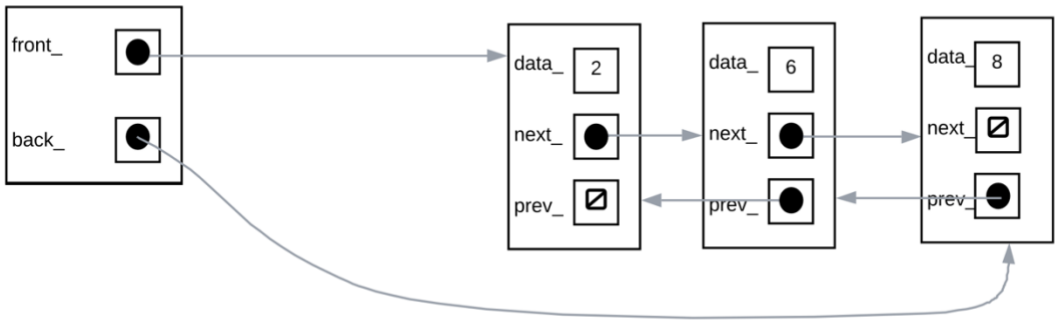
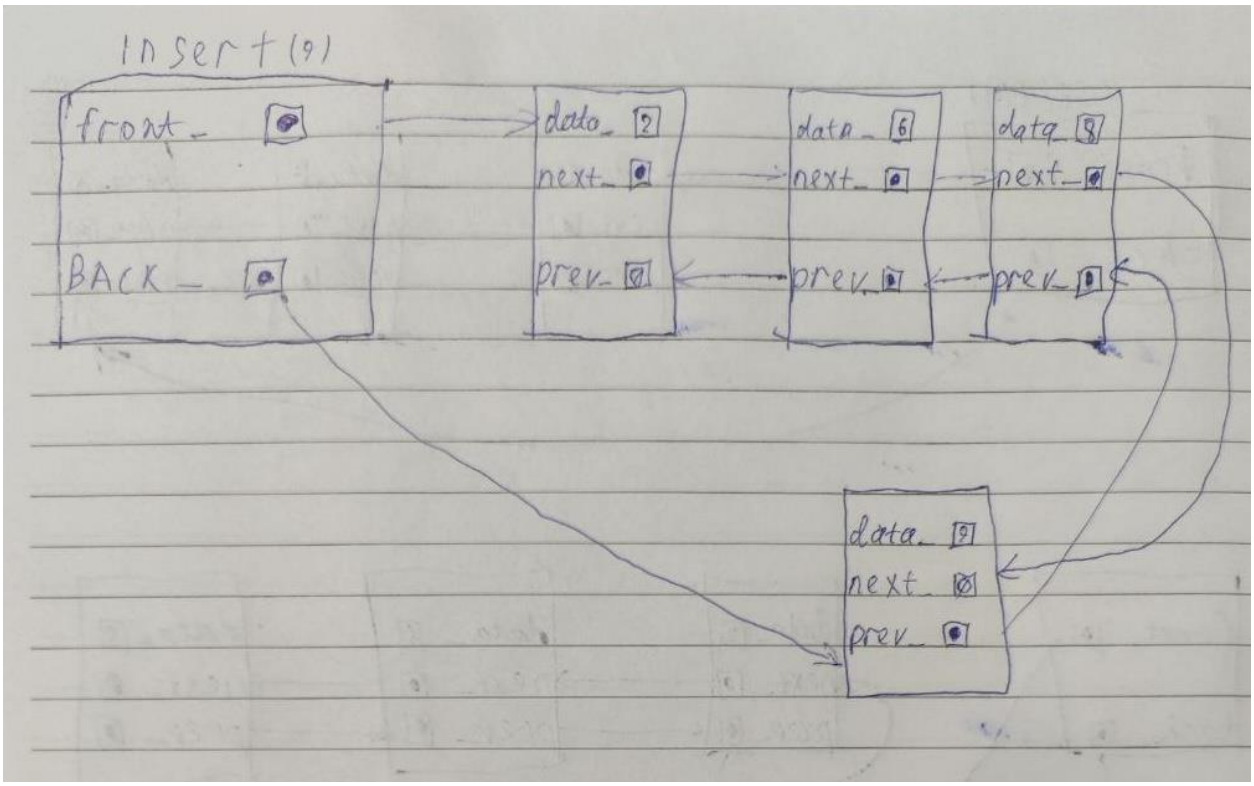
insert(1)


front_
back_
data_ 2
next_
prev_
data_ 6
next_
prev_
data_ 8
next_
prev_

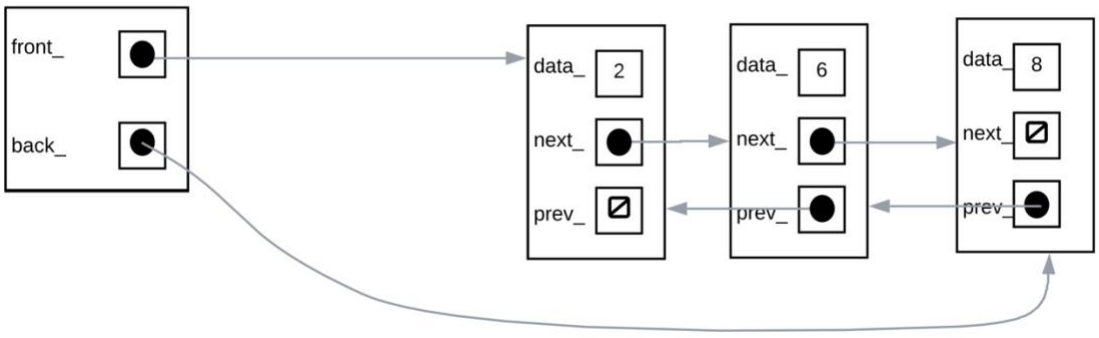

front_
back_
data_ 2
next_
prev_
data_ 6
next_
prev_
data_ 8
next_
prev_
data_ 1
next_
prev_

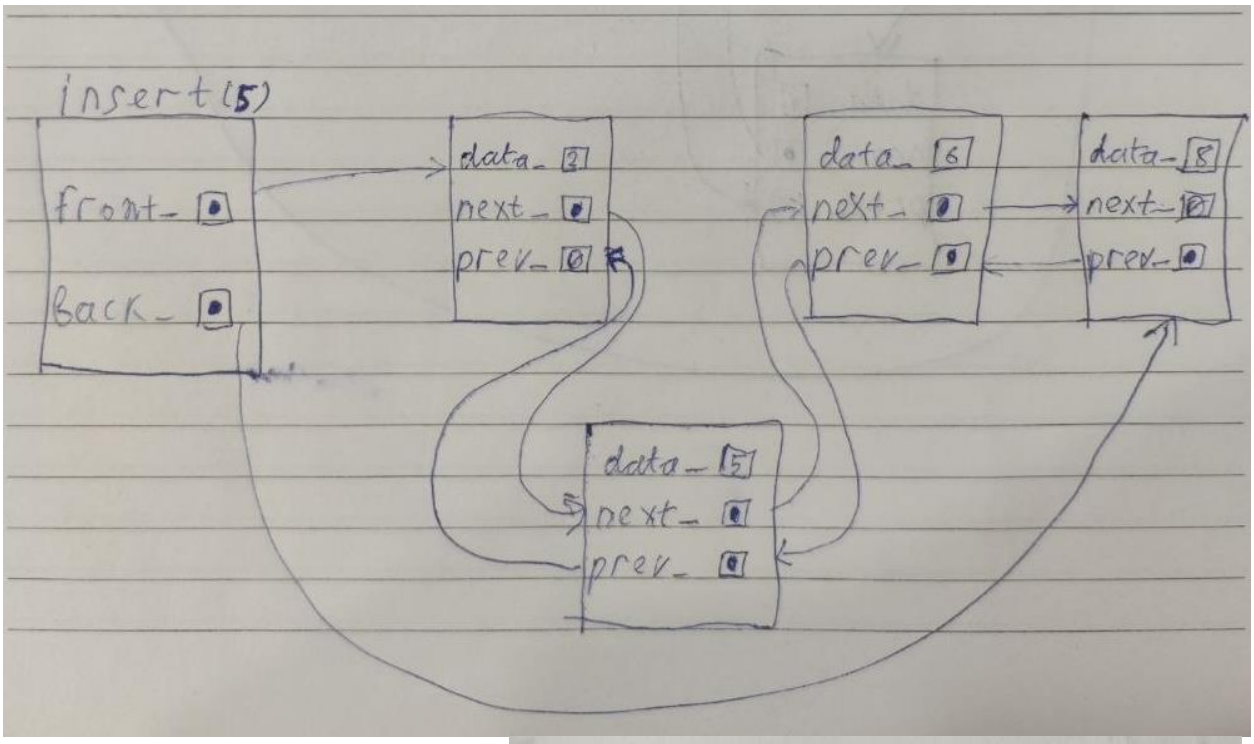
insert(9)



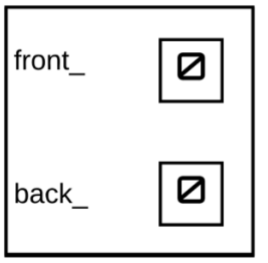


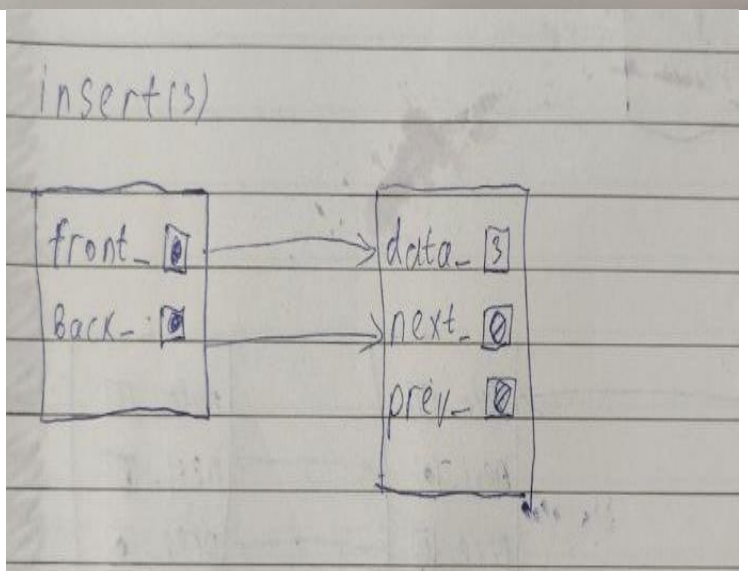
insert(5)



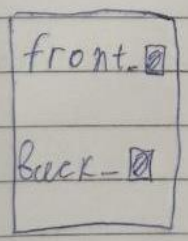

insert(5)
front_
back_
data_ 3
next_
prev_
data_ 6
next_
prev_
data_ 8
next_
prev_
data_ 5
next_
prev_

insert(3)

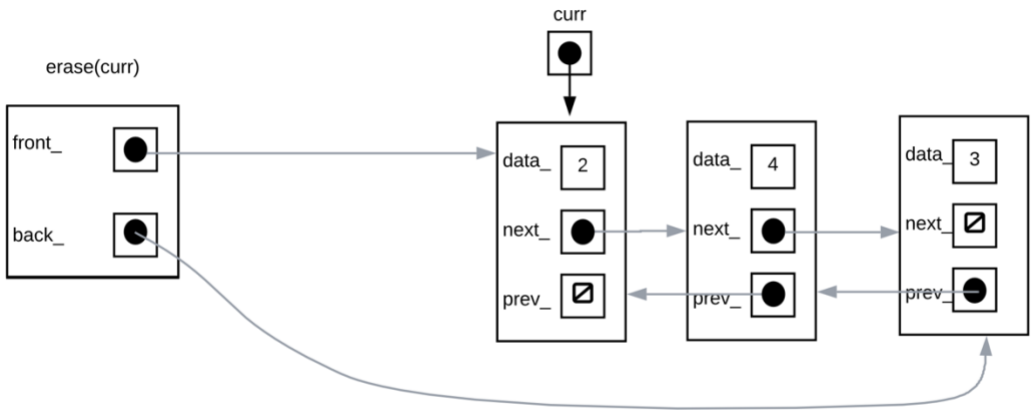

front_
back_


insert(3)
front_
back_
data_ 3
next_
prev_

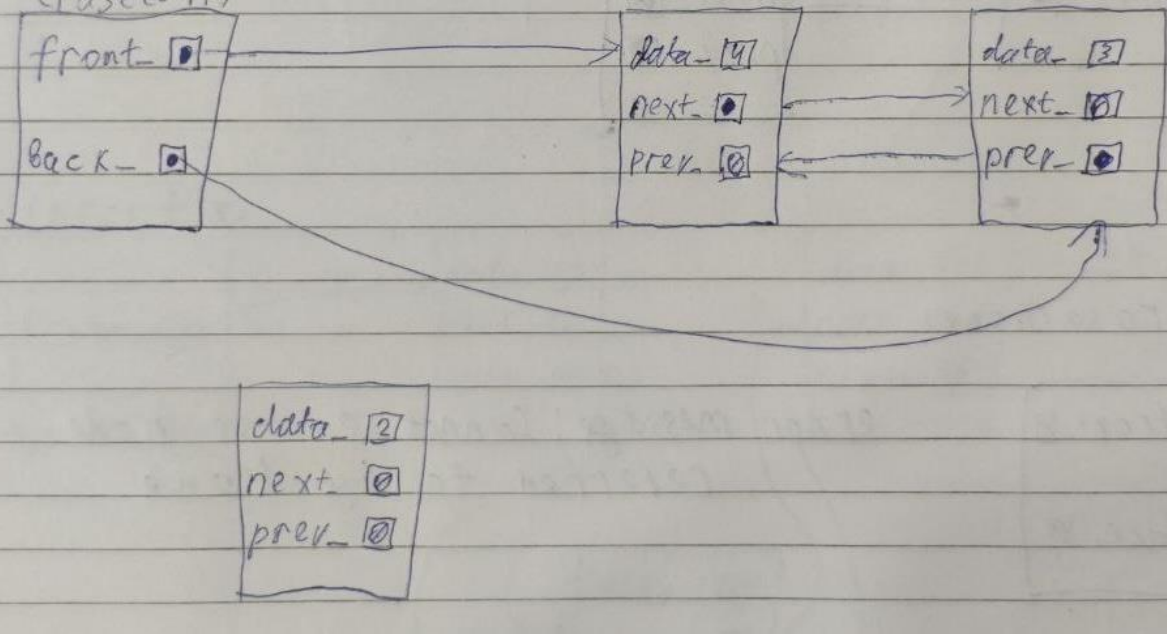
erase(none)

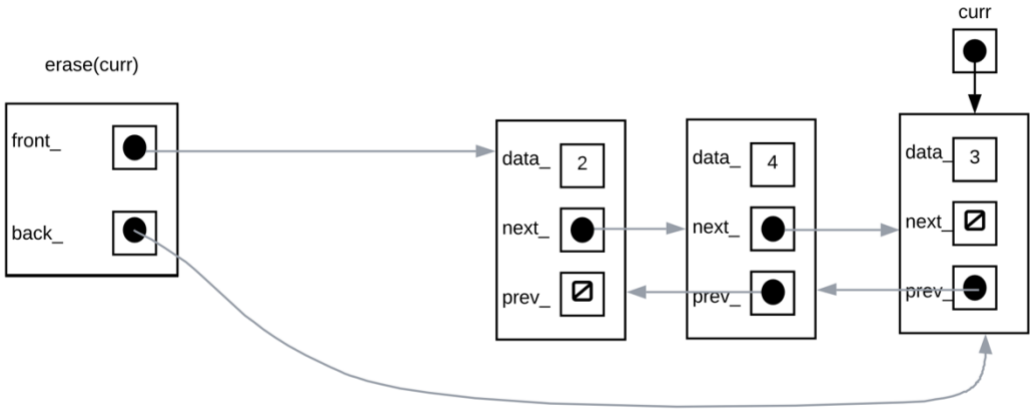


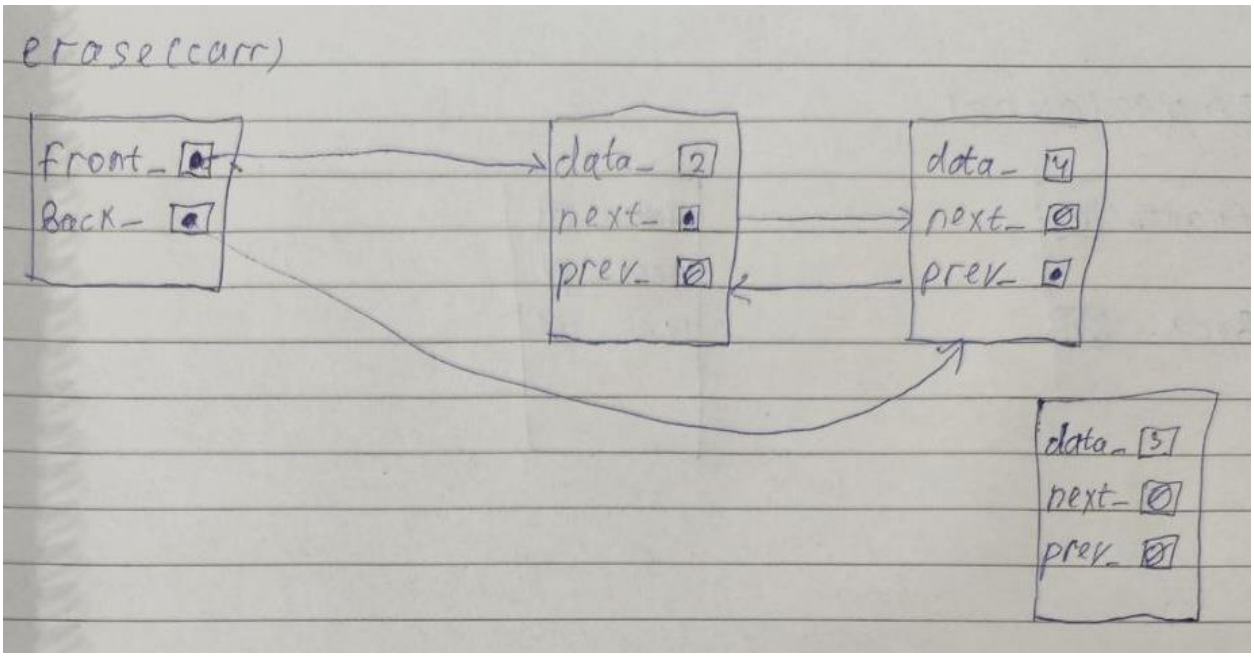
error message: Cannot erase node referred to By None

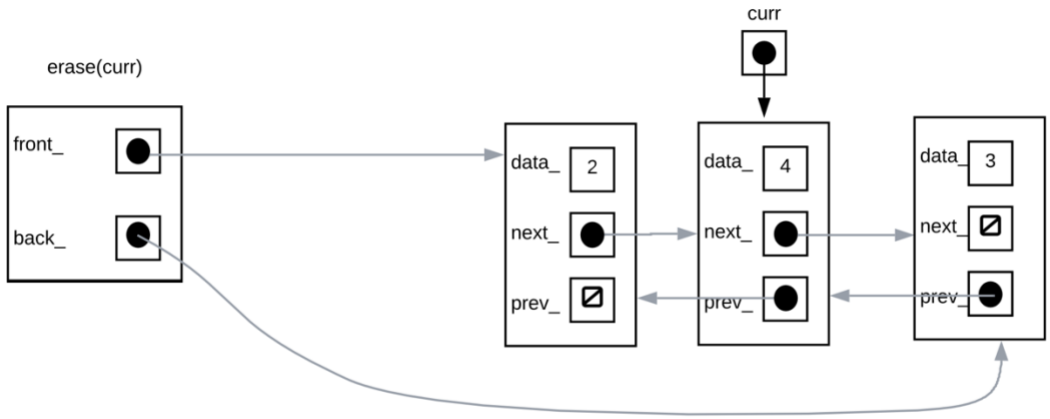


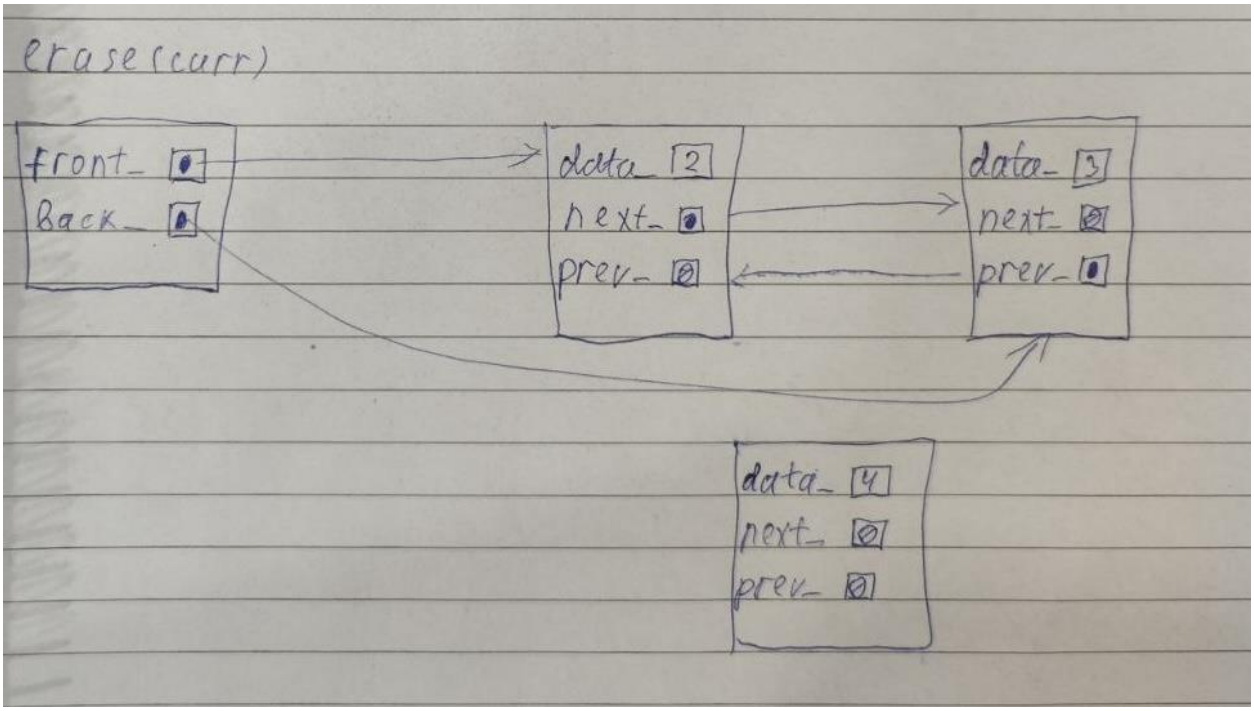
erase(curr)

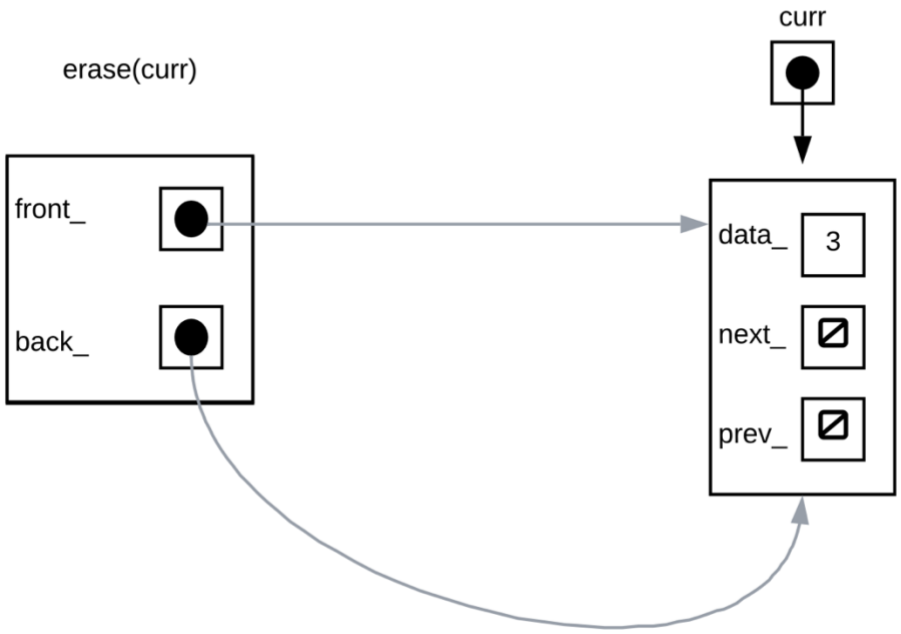


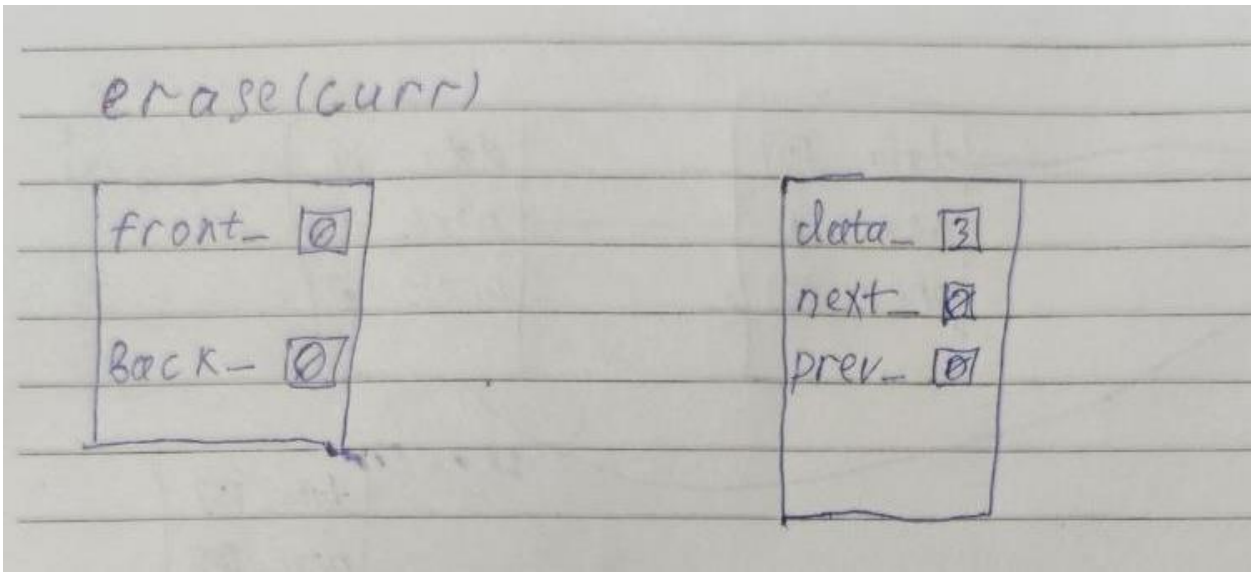

curr
erase(curr)
front_
back_
data_ 2
next_
prev_
data_ 4
next_
prev_
data_ 3
next_
prev_


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


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


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

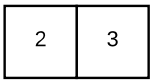

erase(curr)
curr
front_
back_
data_
3
next_
prev_


erase(curr)
front_ 0
Back_ 0
data_ 3
next_ 0
prev_ 0

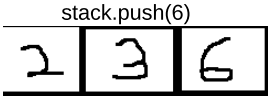
Stack: In the diagrams below list what data members you need to track and what their values are in its initial state and their state after each of the operations are applied to the diagram. If the array needs to be resized, draw the new array with the correct capacity

Data track:
stack: A list representing the stack.
capacity: The current capacity of the stack.
top: The index pointing to the top element (the most recent item).

3 is at top of stack



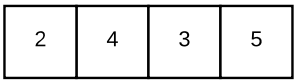
stack.push(6)



Stack = [2, 3,6]
Capacity = 2
Top_val = Index 2

Resize action:
Since the capacity was exceeded, the array is resized: Old array: [2, 3]
New array: [2, 3, 6, None]
(**capacity** is now doubled to 2)

initially 5 is at top of stack



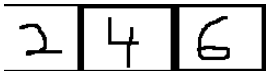
stack.pop()

Stack = [2, 4, 3]
Capacity = 4
Top_val = Index 2

stack.popO

Stack = [2, 4]
Capacity = 4
Top_val = Index 1

stack.push(6)



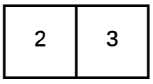
Stack = [2, 4,6]
Capacity = 4
Top_val = Index 2

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

First-In-First-Out (FIFO)
The front is where elements are removed, and the back is where elements are added.

Data Members to Track:
Queue (array): This is the list that holds the elements in the queue.
Capacity: This keeps track of the current size limit of the queue.
Front: Index of the element at the front of the queue
Back: Index of the element at the back of the queue
Size: Tracks the number of elements in the queue (since the queue might not be full even if the back index has reached the capacity, due to dequeuing).

2 is at front of queue, 3 is at back

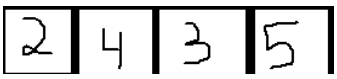


queue.enqueue(6)

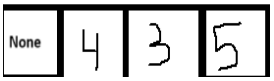
Queue: [2,3, 6] (added 6 to the back of the queue)
Front: Index 0 (pointing to 2)
Back: Index 2 (pointing to value 6)
Capacity: needs to be doubled (2 ->48)

Resizing
New array: [2,3, 6, None]
(capacity doubled to 4)

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

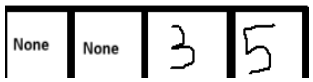


queue.dequeue()



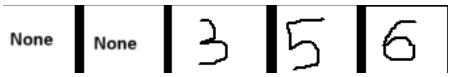
Queue: [None, 4,3, 5] (removed 2 from the front)
Front: Index 1(pointing to value 4)
Back: Index 3 (pointing to 5)
Capacity: 4(remains the same)

queue.dequeue()



Queue: [None, None,3, 5] (removed 4 from the front)
Front: Index 2(pointing to value 3)
Back: Index 3 (pointing to 5)
Capacity: 4(remains the same)

queue.enqueue(6)

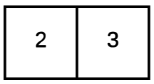


Queue: [None, None,3, 5,6] (added 6 from the back)
Front: Index 2(pointing to value 3)
Back: Index 4 (pointing to 6)
Capacity: 4(remains the same)

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

Data Members to Track:
Deque (array): The list that holds the elements in the deque.
Capacity: The current size limit of the deque.
Front: Index of the front element (where elements can be added or removed).
Back: Index of the back element (where elements can be added or removed).
Size: The current number of elements in the deque.

2 is at front of queue, 3 is at back

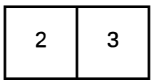


deque.push_front(6)

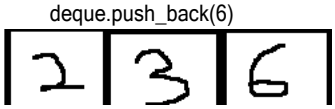
Deque: [2, 3,6] (added 6 to the front)
Front: Index 0 (pointing to value 6)
Back: Index 2 (pointing to value 3)
Capacity: needs to be doubled (2 -> 4)
Size: 3(one element was added)

Resizing
New array: [6,2,3,None]
(capacity doubled to 4)

2 is at front of queue, 3 is at back



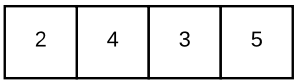
deque.push_back(6)



Deque : [2,3,6] (6 is added to the back)
Capacity: 4
Front: 0 (element 2 is at the front)
Back: 2 (element 6 is at the back)
Size: 3 (one element was added)

Resizing
New array: [2,3,6,None]
(capacity doubled to 4)

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



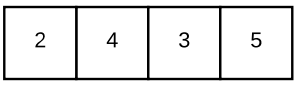
deque.pop_back()

Deque : [2,4 3] (5 is removed to the back)
Capacity: 4
Front: 0 (element 2 is at the front)
Back: 3 (element 3 is at the back)
Size: 3 (one element was removed)

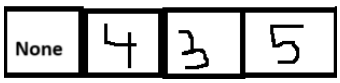
deque.push_front(6)

Deque : [6,2,4 3] (6 is added to the front)
Capacity: 4
Front: 0 (element 6 is at the front)
Back: 3 (element 3 is at the back)
Size: 4 (one element was added)

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



deque.pop_front()



Deque: [None, 4, 3, 5] (removed 2)
Capacity: 4
Front: 1 (element 4 is now at the front)
Back: 3 (element 5 is still at the back)
Size: 3 (one element was removed)

deque .push_back(6)

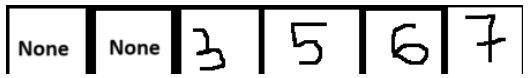
Deque: [None, 4, 3, 5, 6 (added 6 from the back)
Capacity: 8
Front: 1 (element 4 is now at the front)
Back: 4 (element 6 is at the back)
Size: 4(one element was added)

Resizing Newarray: [None,2,3,5,6] (capacitydoubledto8)

deque.pop_front()

Deque: [None, None, 3, 5,6]
(removed 4 from front)
Capacity: 8
Front: 2 (4 is now at the front)
Back: 3 (6 is still at the back)
Size: 3 (one element was removed)

deque.push_back(7)



Deque: [None, None, 3, 5, 6,7 (added 7 from the back)
Capacity: 8
Front: 2 (3 is now at the front)
Back: 5 (7 is at the back)
Size: 4(one element was added)

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

-

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

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

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

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

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

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