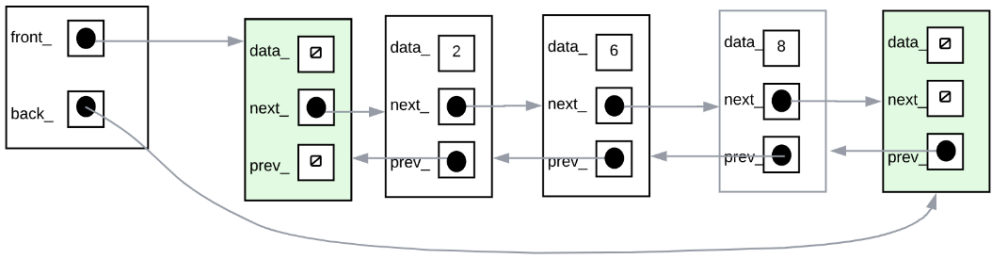
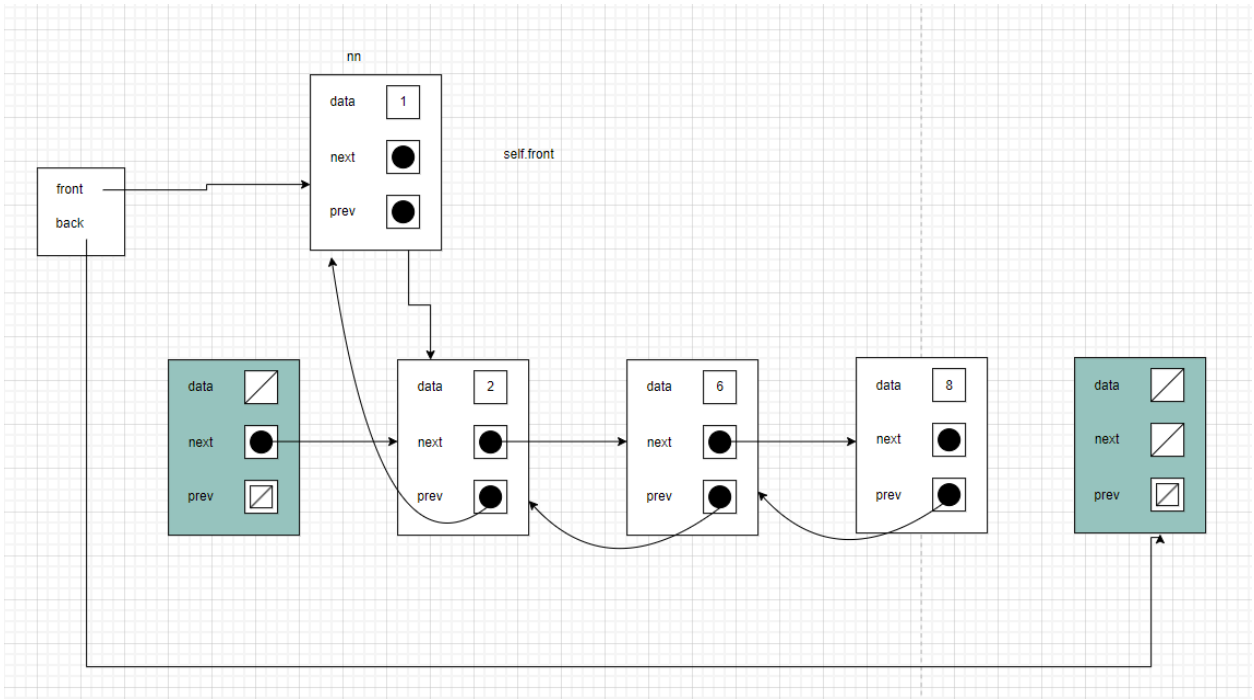
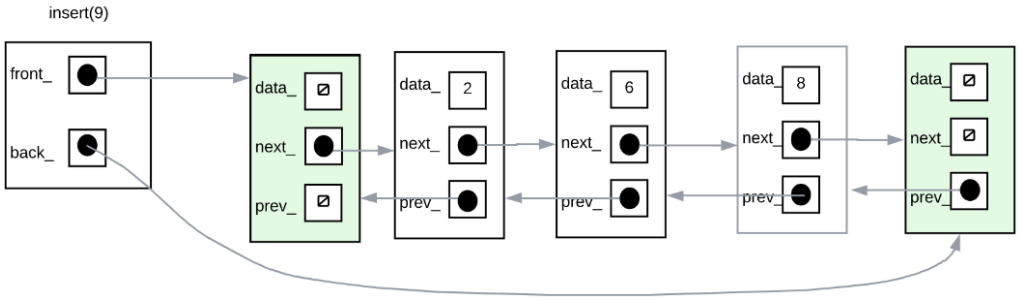
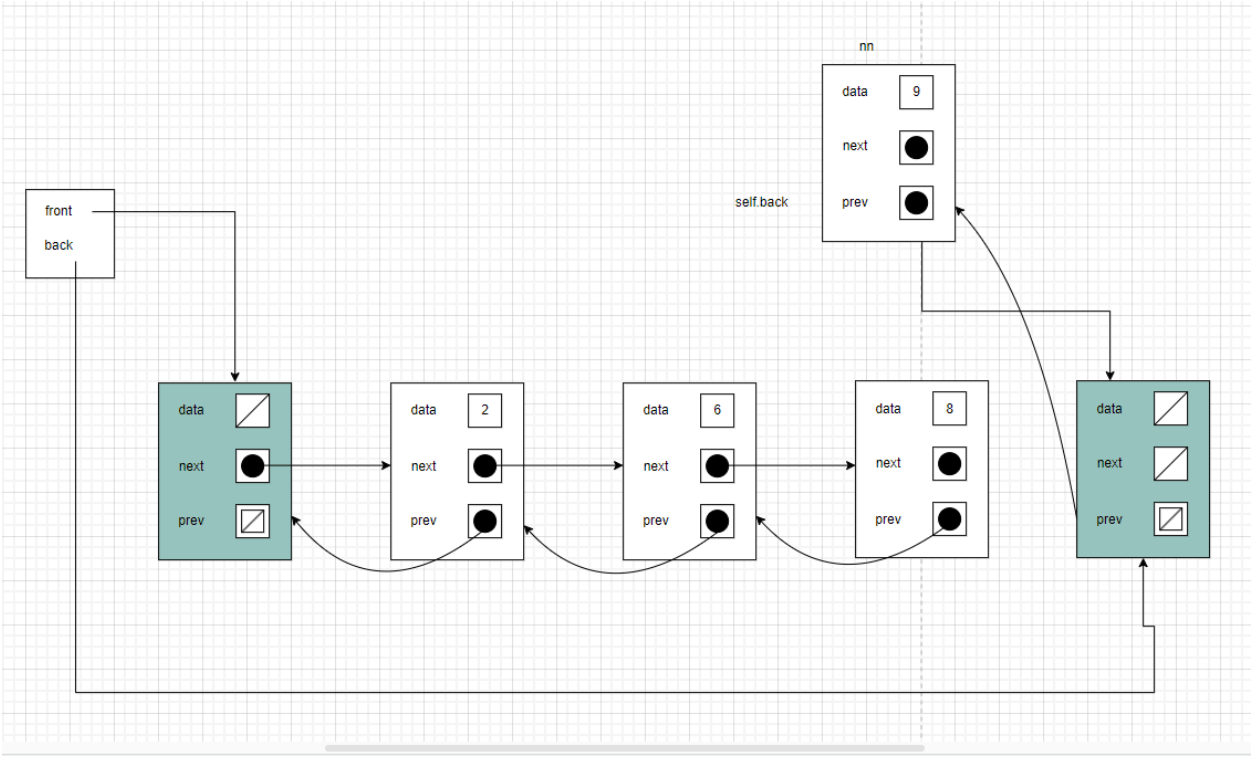
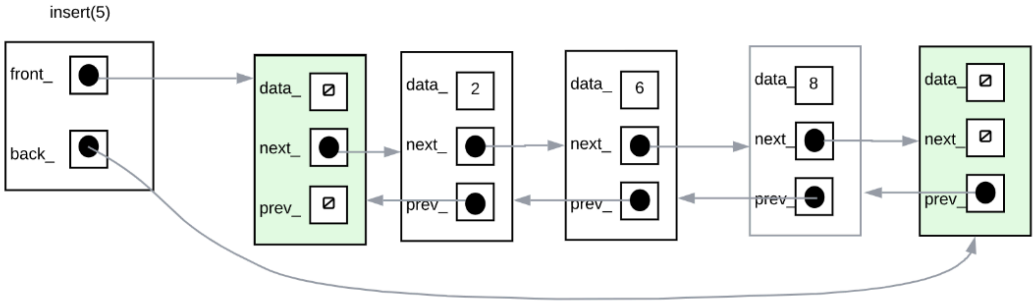
insert(1)

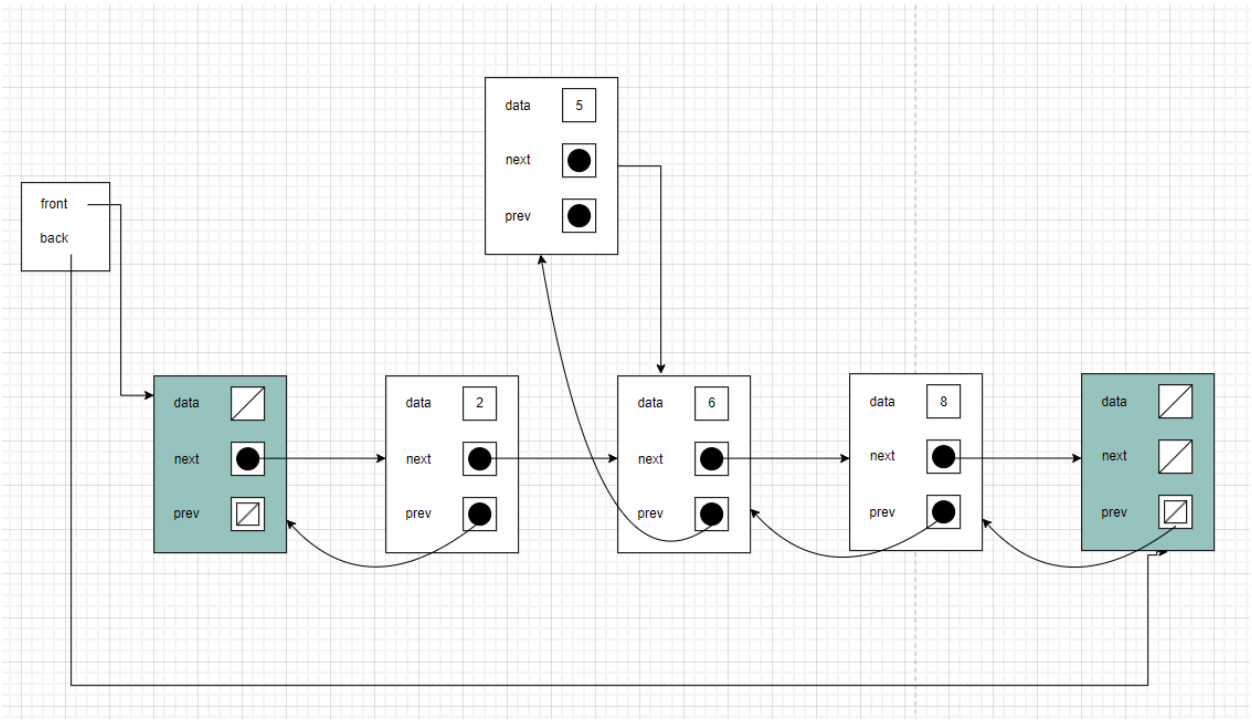

front_
back_
data_
next_
prev_
data_ 2
next_
prev_
data_ 6
next_
prev_
data_ 8
next_
prev_
data_
next_
prev_

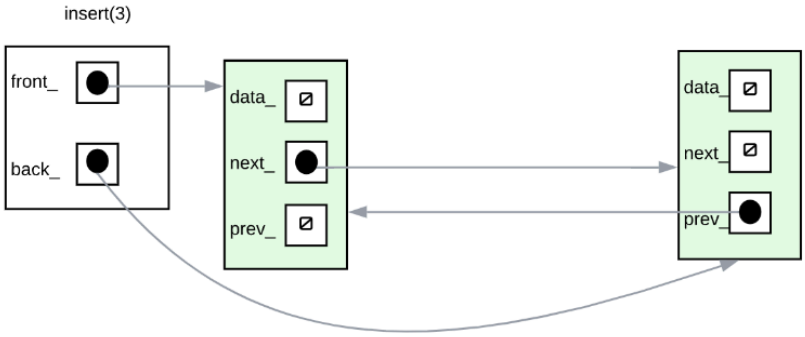

nn
data 1
next
prev
self.front
front
back
data
next
prev
data 2
next
prev
data 6
next
prev
data 8
next
prev
data
next
prev

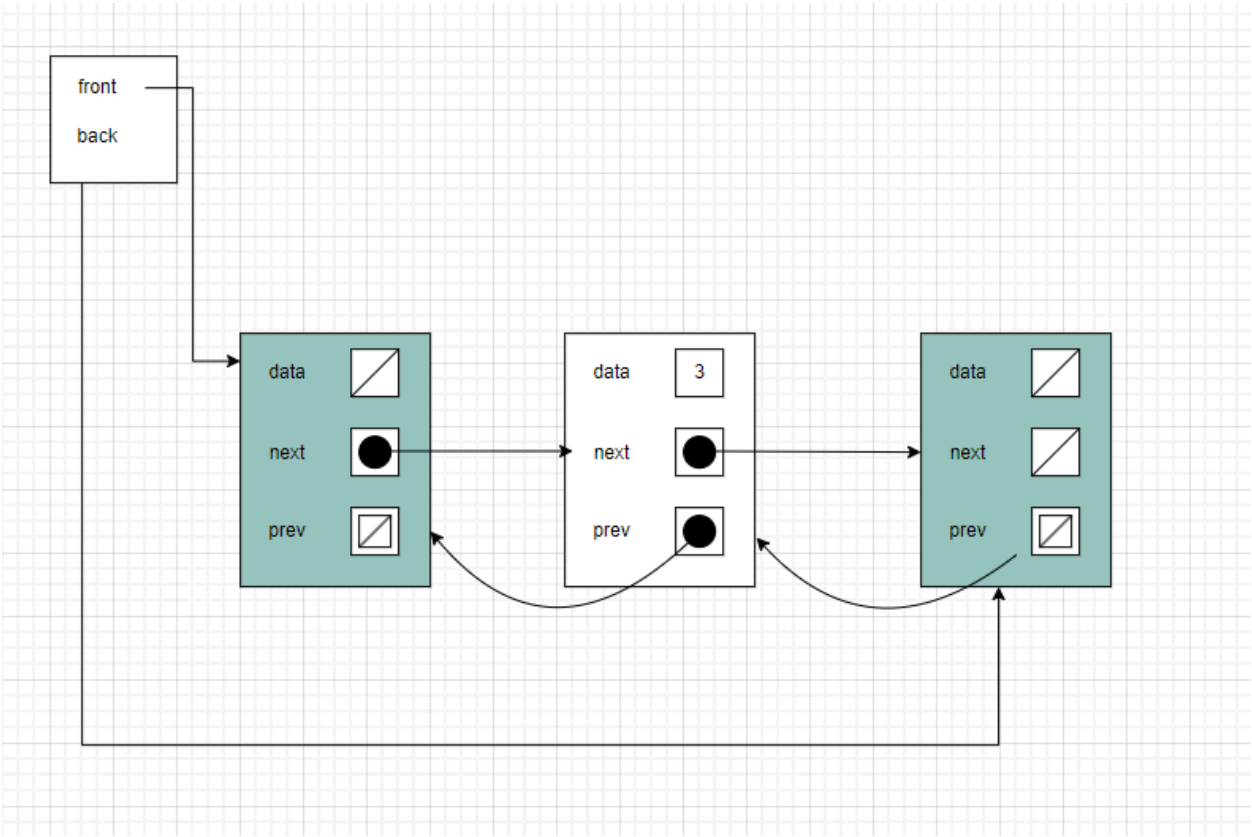

insert(9)
front_
back_
data_
next_
prev_
data_ 2
next_
prev_
data_ 6
next_
prev_
data_ 8
next_
prev_
data_
next_
prev_

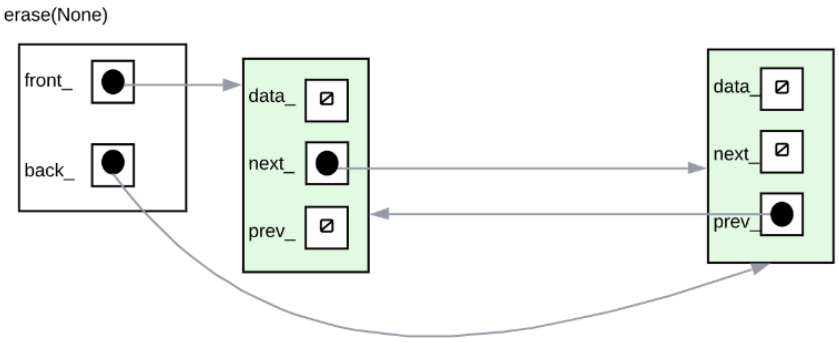

nn
data 9
next
prev
self.back
front
back
data
next
prev
data 2
next
prev
data 6
next
prev
data 8
next
prev
data
next
prev

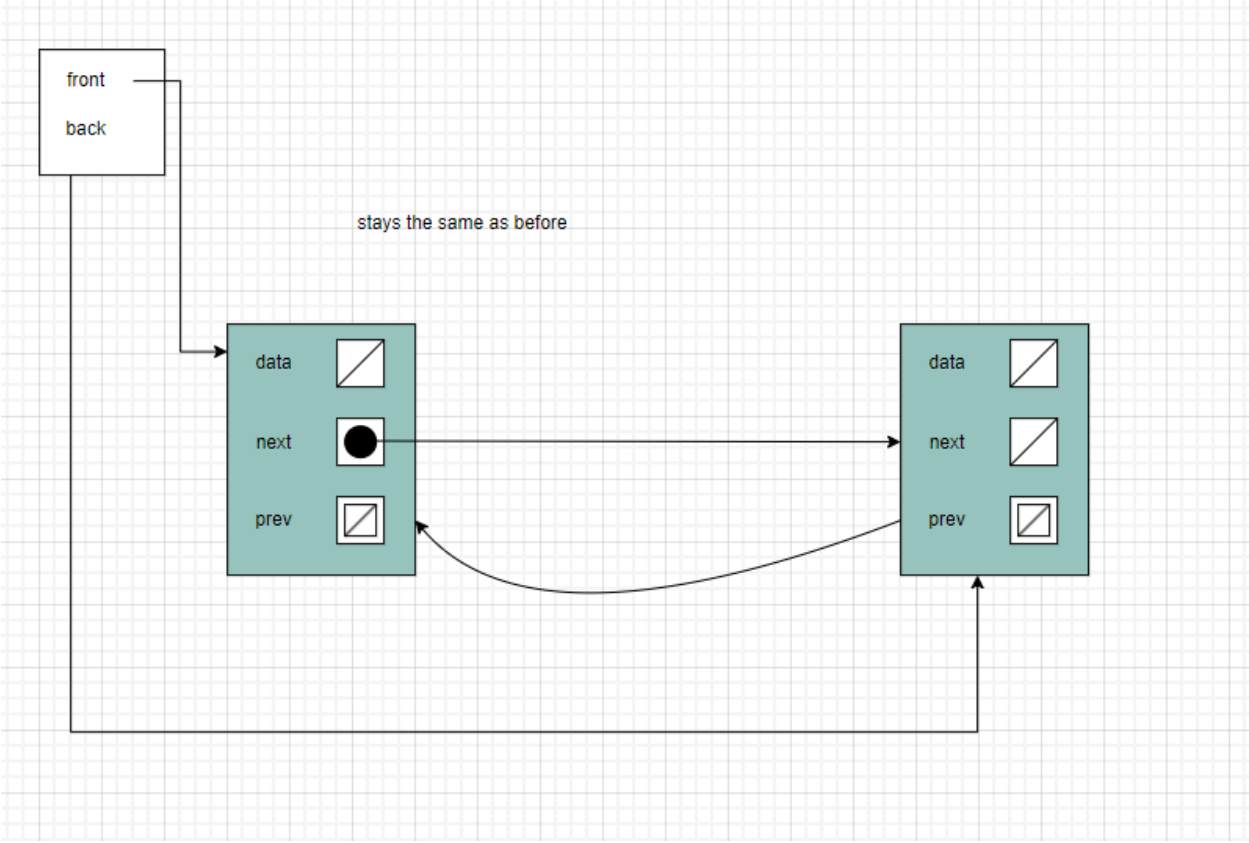

insert(5)
front_
back_
data_
next_
prev_
data_ 2
next_
prev_
data_ 6
next_
prev_
data_ 8
next_
prev_
data_
next_
prev_

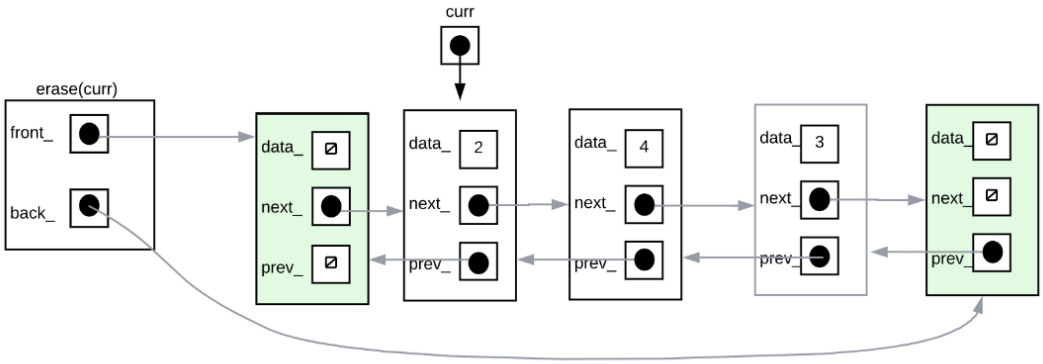

front
back
data 5
next
prev
data
next
prev
data 2
next
prev
data 6
next
prev
data 8
next
prev
data
next
prev

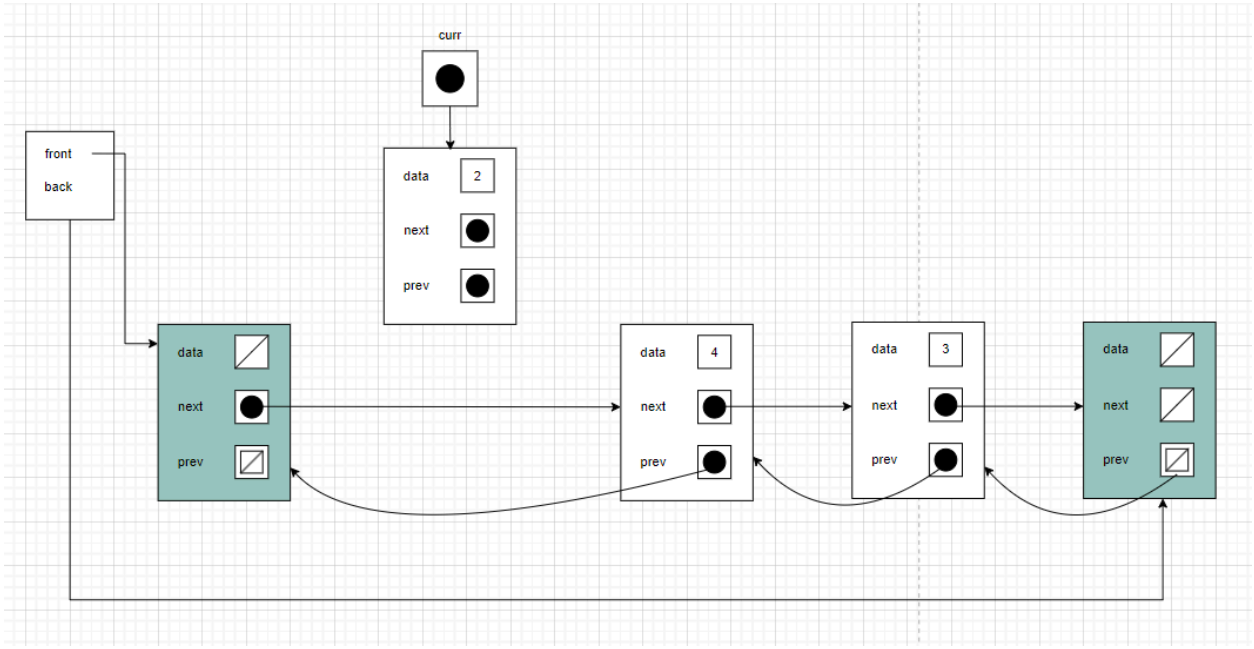

insert(3)
front_
back_
data_
next_
prev_
data_
next_
prev_

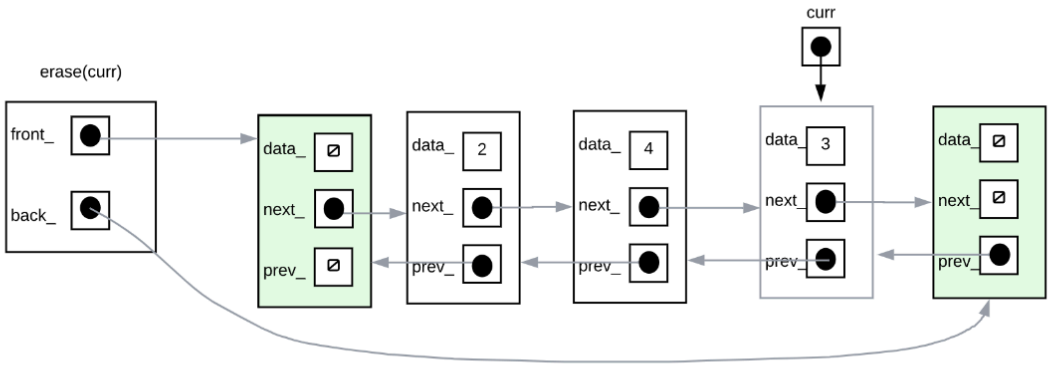

front
back
data
next
prev
data
3
next
prev
data
next
prev

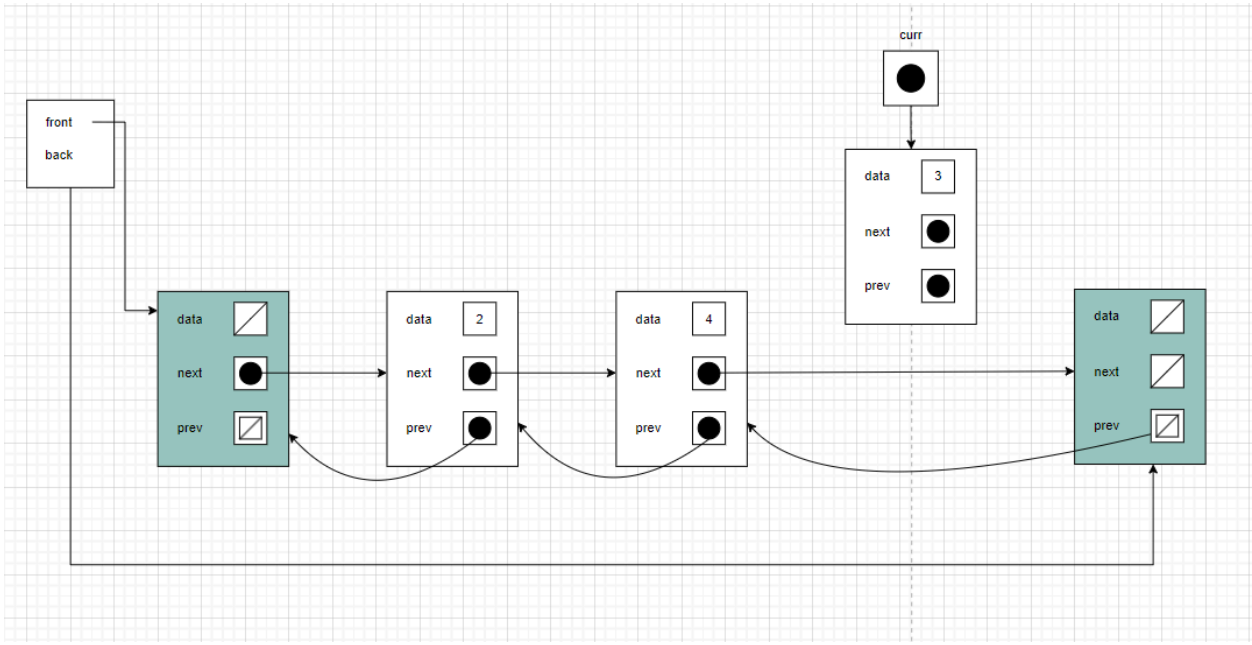

erase(None)
front_
back_
data_
next_
prev_
data_
next_
prev_

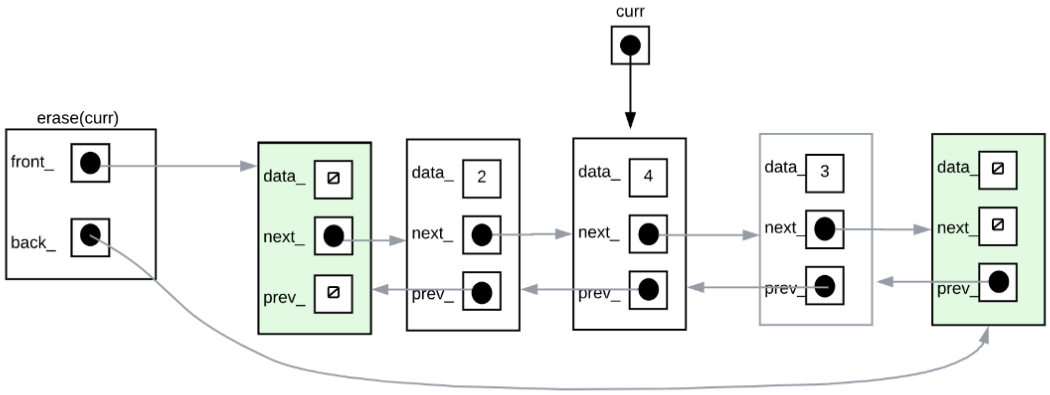

front
back
stays the same as before
data
next
prev
data
next
prev

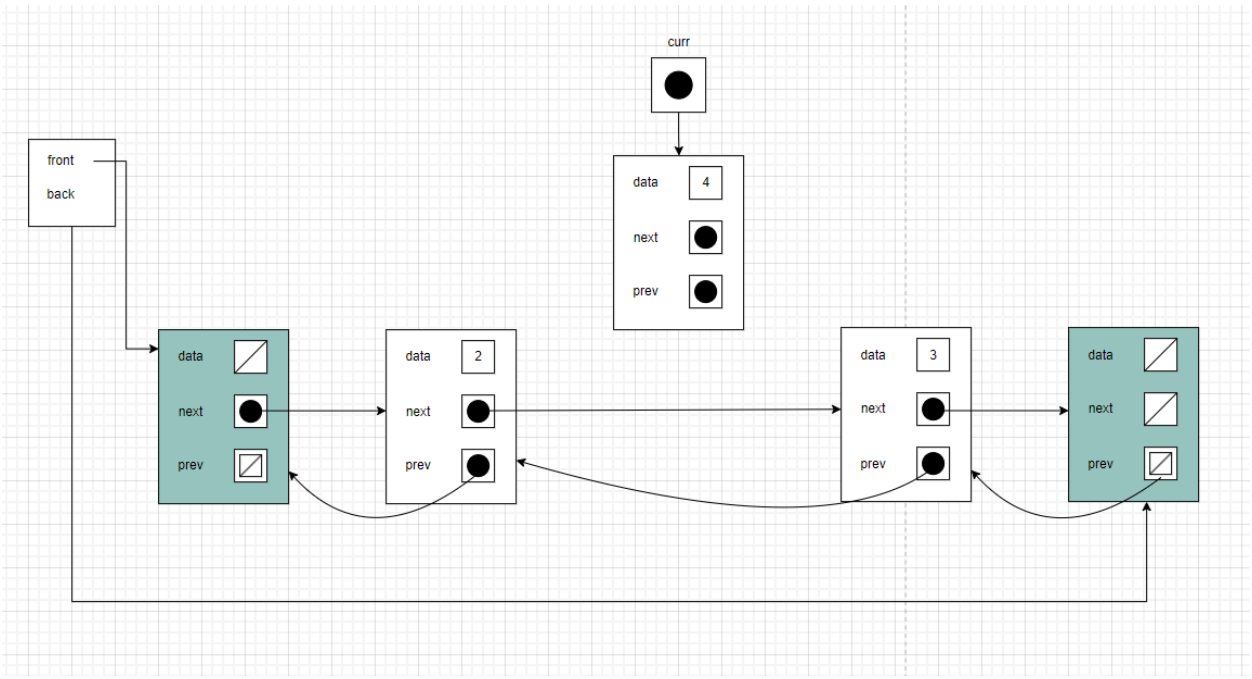

curr
erase(curr)
front_
back_
data_
next_
prev_
2
4
3

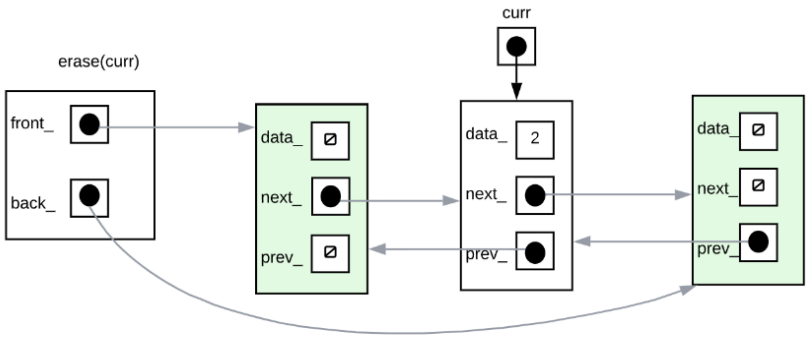

curr
front
back
data
next
prev
2
4
3

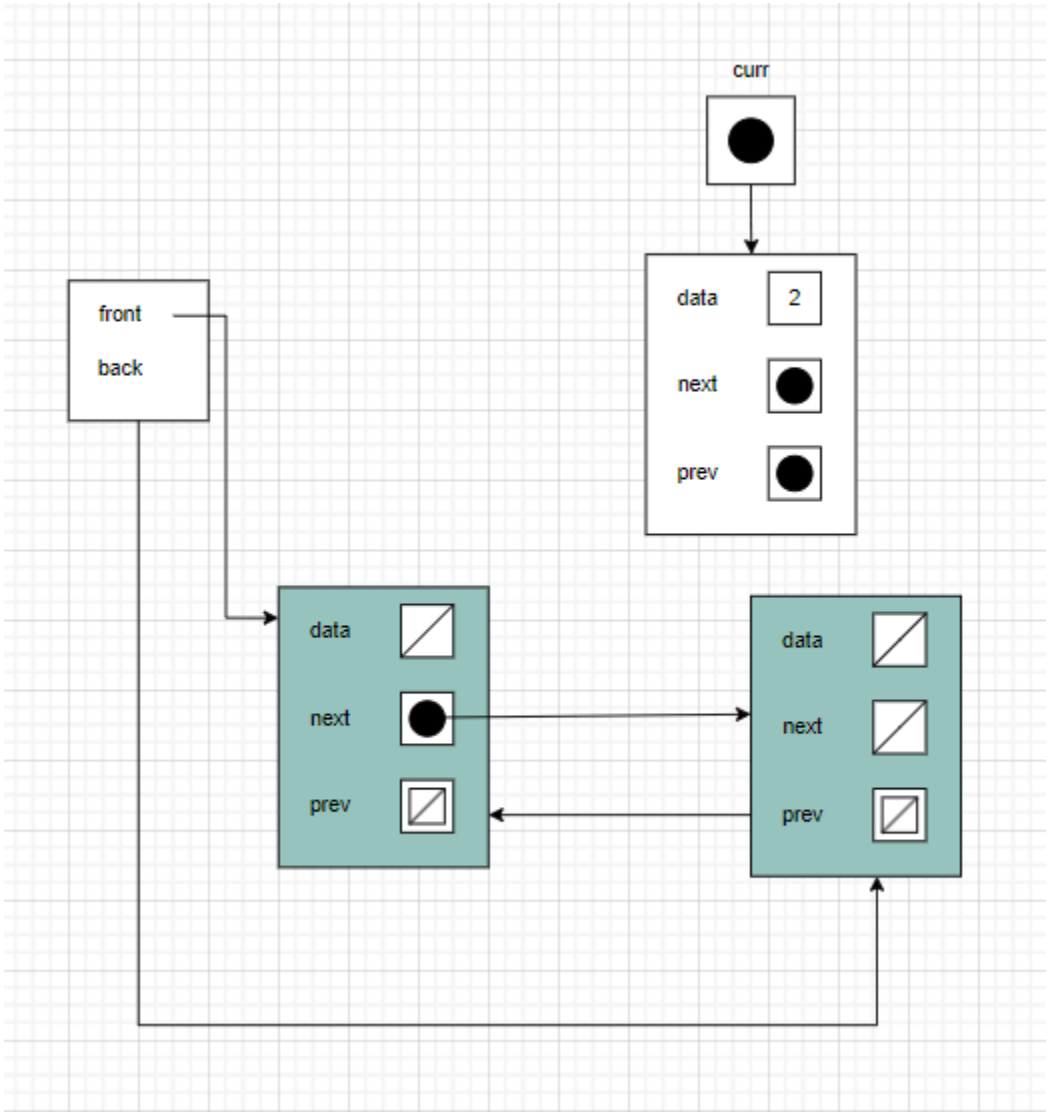

curr
erase(curr)
front_
back_
data_
next_
prev_
data_ 2
next_
prev_
data_ 4
next_
prev_
data_ 3
next_
prev_
data_
next_
prev_


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
front
back
data 4
next
prev
data
next
prev
data 2
next
prev
data 3
next
prev
data
next
prev


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

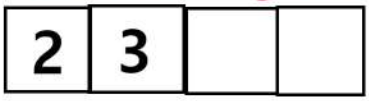

curr
front
back
data
2
next
prev
data
next
prev
data
next
prev

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

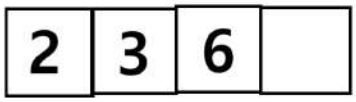
stack.push(6)

3 is at top of stack


**After resizing to 4**

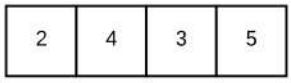



**After stack.push(6)**




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

initially 5 is at top of stack




**After first stack.pop()**


**After second stack.pop()**


**After stack.push(6)**

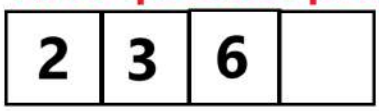
Queues: In the diagrams below list what data members you need to track and what their values are in its initial state and their state after each of the operations are applied to the diagram. If the array needs to be resized, draw the new array with the correct capacity

queue.enqueue(6)

2 is at front of queue, 3 is at back

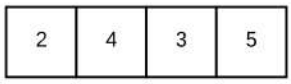

**After resizing to 4**


**After queue.enqueue(6)**



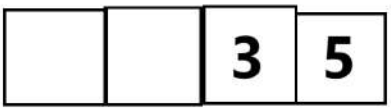

queue.dequeue()
queue.dequeue()
queue.enqueue(6)

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




**After first queue.dequeue()**


**After second queue.dequeue()**

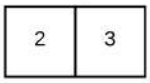



**After queue.enqueue(6)**

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

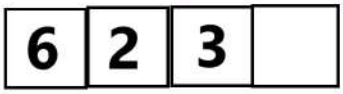
deque.push_front(6)

2 is at front of Deque, 3 is at back



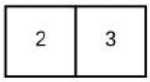

**After resizing to 4**


**After deque.push_front(6)**




deque.push_back(6)

2 is at front of Deque, 3 is at back

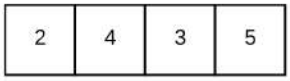



**After resizing to 4**

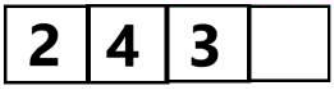

**After deque.push_back(6)**


deque.pop_back()
deque.push_front(6)

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



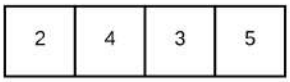

**After deque.pop_back()**




**After deque.push_front(6)**

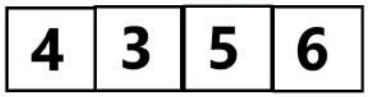

deque.pop_front()
deque.push_back(6)
deque.pop_front()
deque.push_back(7)

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



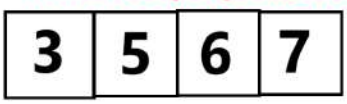

**1. After deque.pop_front()**


**2. After deque.push_back(6)**




**3. After deque.pop_front()**


**4. After deque.push_back(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