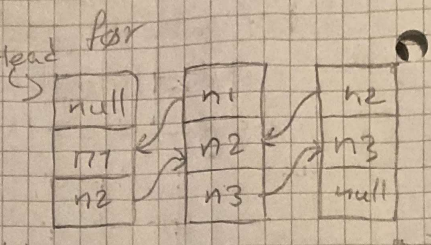
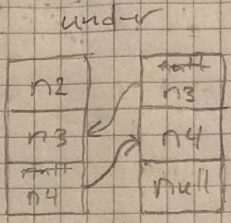
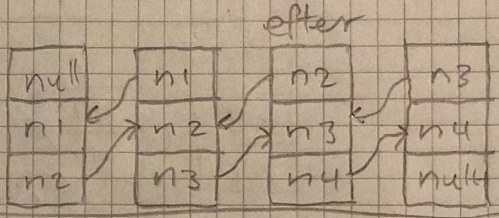
Doubly Linked List

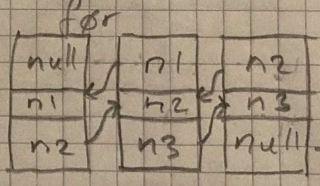
```
addLast(data) {
    if (tail = null)
        tail = new node(data)
        head = "
        size++
    else
        tail.next = new node(data)
        new node(data).prev = tail
        tail = new node(data)
        size ++
}
```

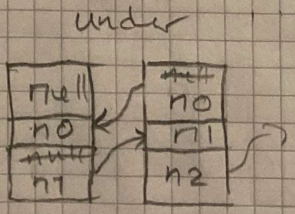


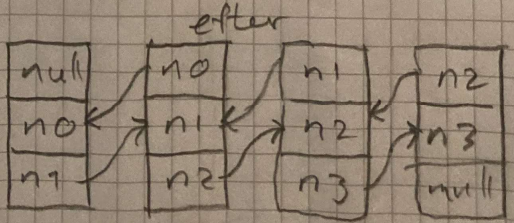




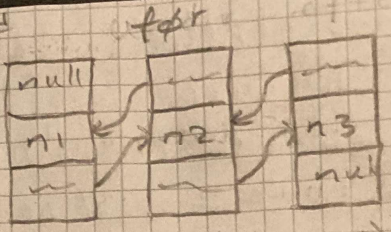

```
addFirst(data) {
    if (head = null)
        head = nynode(data)
        tail = nynode(data
    else
        head.prev = nynode(data)
        nynode.next = head
        head = nynode
```



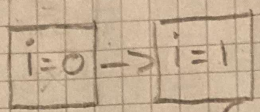




```
get (index) {
    if (index outofBounds)
        return null
    node = head
    for (i < index, i++)
        node = node.next for (i)
    return node når for loop
    når t.l index som
    er den ønskede node
```



under (index = 1)

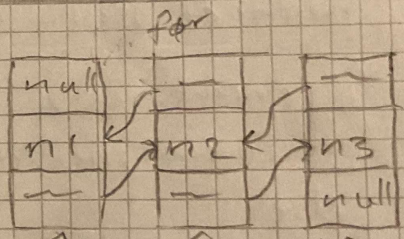


x

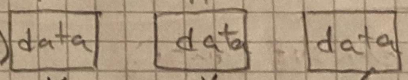
efter

return n2

```
indexOf (data)
    node = head
    index = 0
    while (node != null)
        if (node.data == data)
            return index
        node = next
        index ++
```



indexOf(n3)



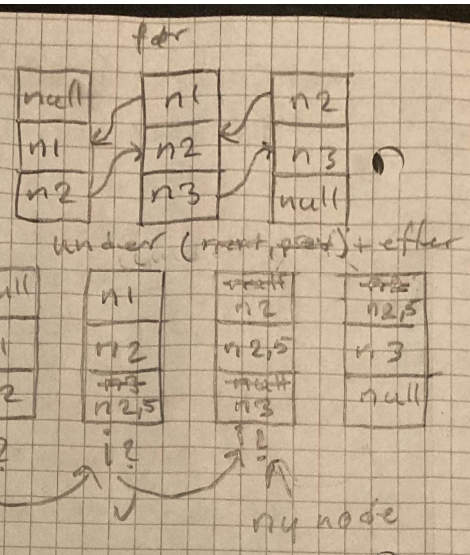
under

x x ✓

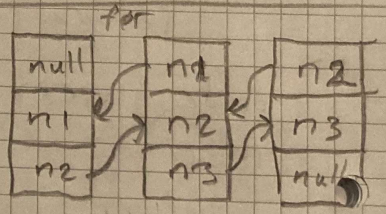
efter

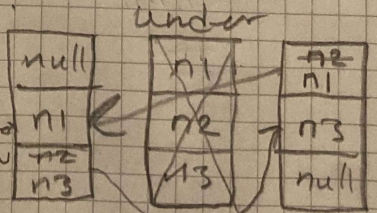
undret listen
men returneret
antal iterationer
(index) eller null/-1

```
insertAfter(index, data) {
  if (index out of bounds)
    return null
  node = new node(data)
  current = head
  for (i=0; i<index; i++){
    current = current.next
  }
  newnode.next = current.next
  newnode.prev = current
  if (current.next)
    current.next.prev = newnode
  else
    this.tail = newnode
```

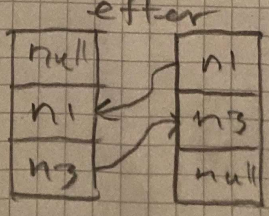


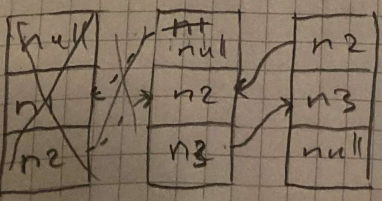

```
remove(data)
  if (head = null) (listen er tom
    return null
  current = this.head
  while (current)
    if (current == data)
      current.prev.next = current.next
      current.next.prev = current.prev
```



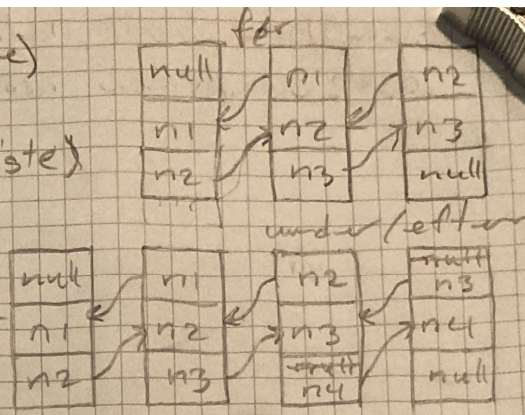


ydarligere if statement der tjekker om current er head eller tail

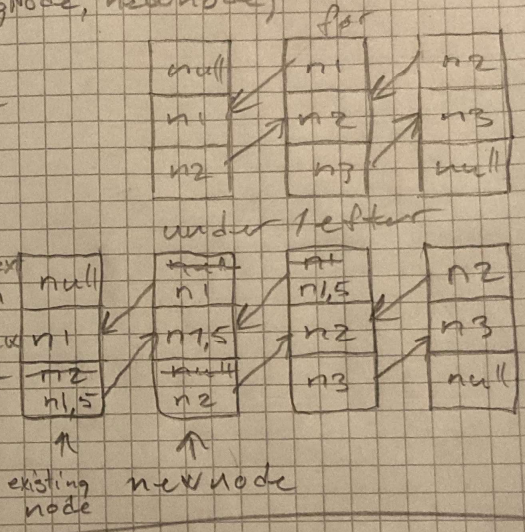




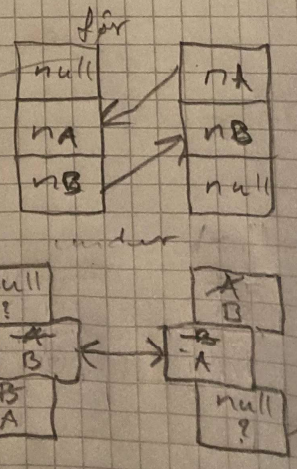
```
addNodeLast(newNode)
  if (head == null)
    return null (tom liste)
  else
    newNode.Prev = tail
    tail.next = newNode
    tail = newNode
```




```
insertAfterNode(existingNode, newNode)
  if (existingNode == tail)
    tail.next = newNode
    newNode.prev = tail
    tail = newNode
  else
    newNode.next = exNod.next
    newNode.prev = existign
    existingNode.Prev.next = new
    existing.next = newNode
```

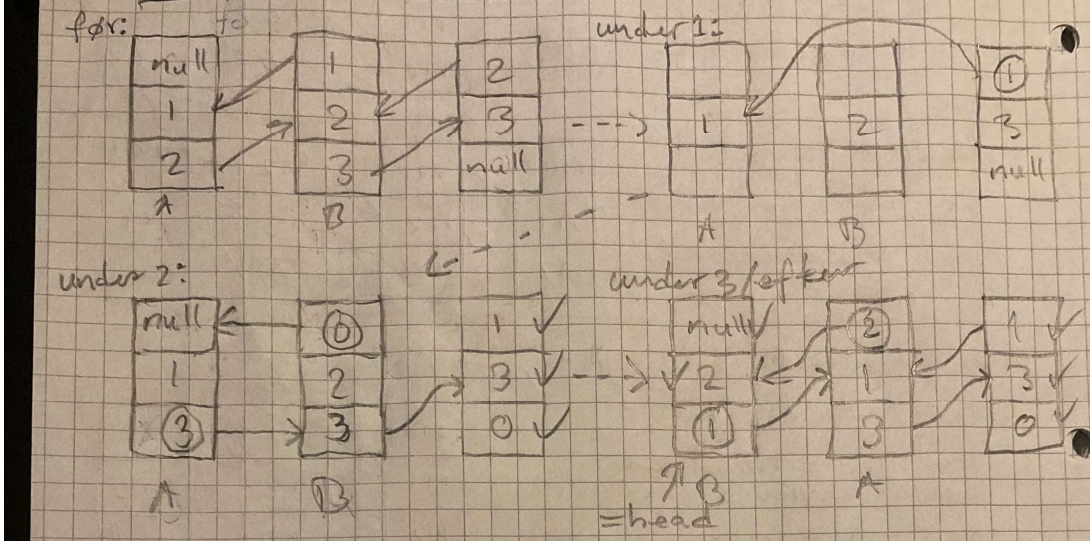



```
swapNodes(nodeA, nodeB)
  const tempData = nodeA.data
  nodeA.data = nodeB.data
  nodeB.data = tempData
```



swapNodes (nodA, nodeB)

case: swapNodes Head



case: swapnodes midt i listen