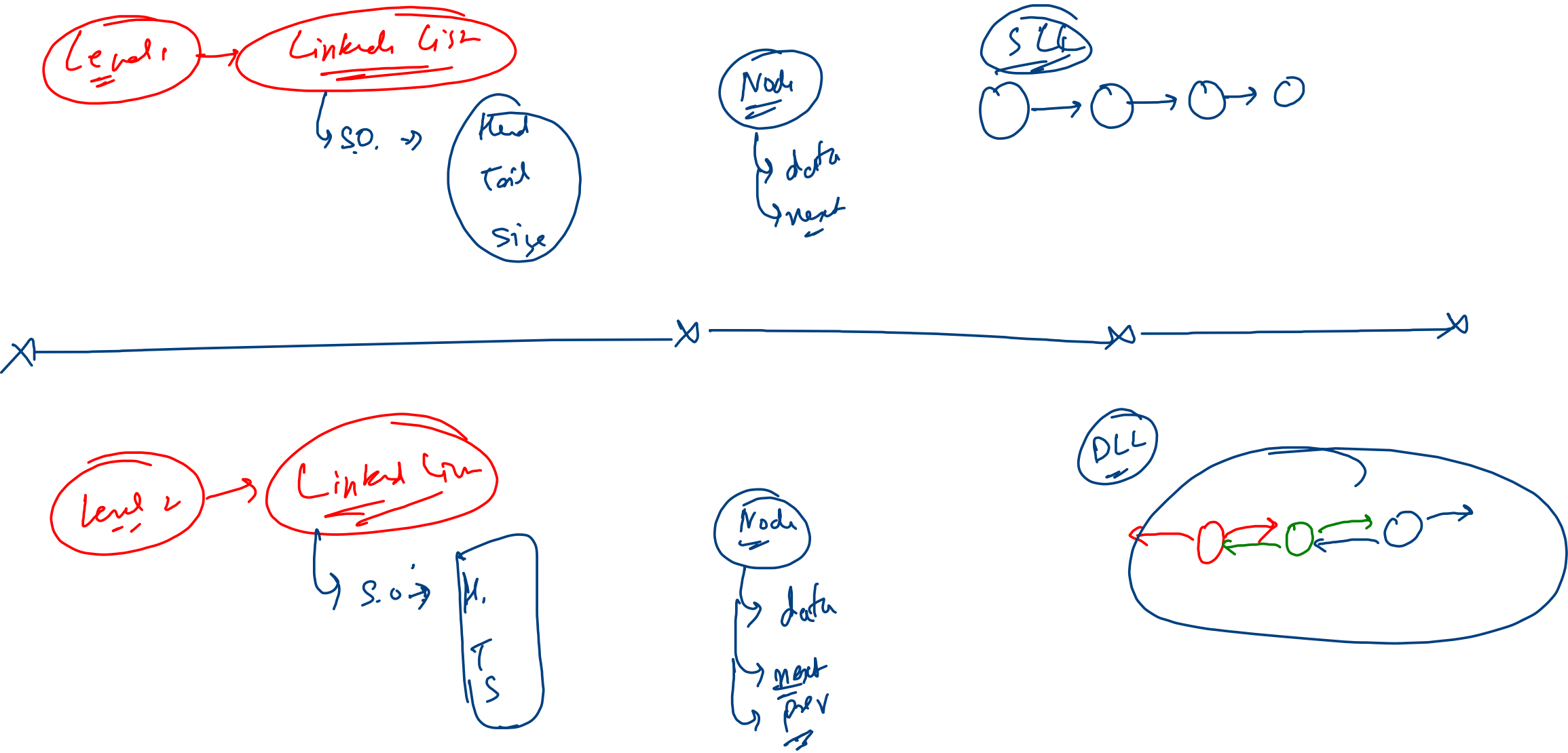
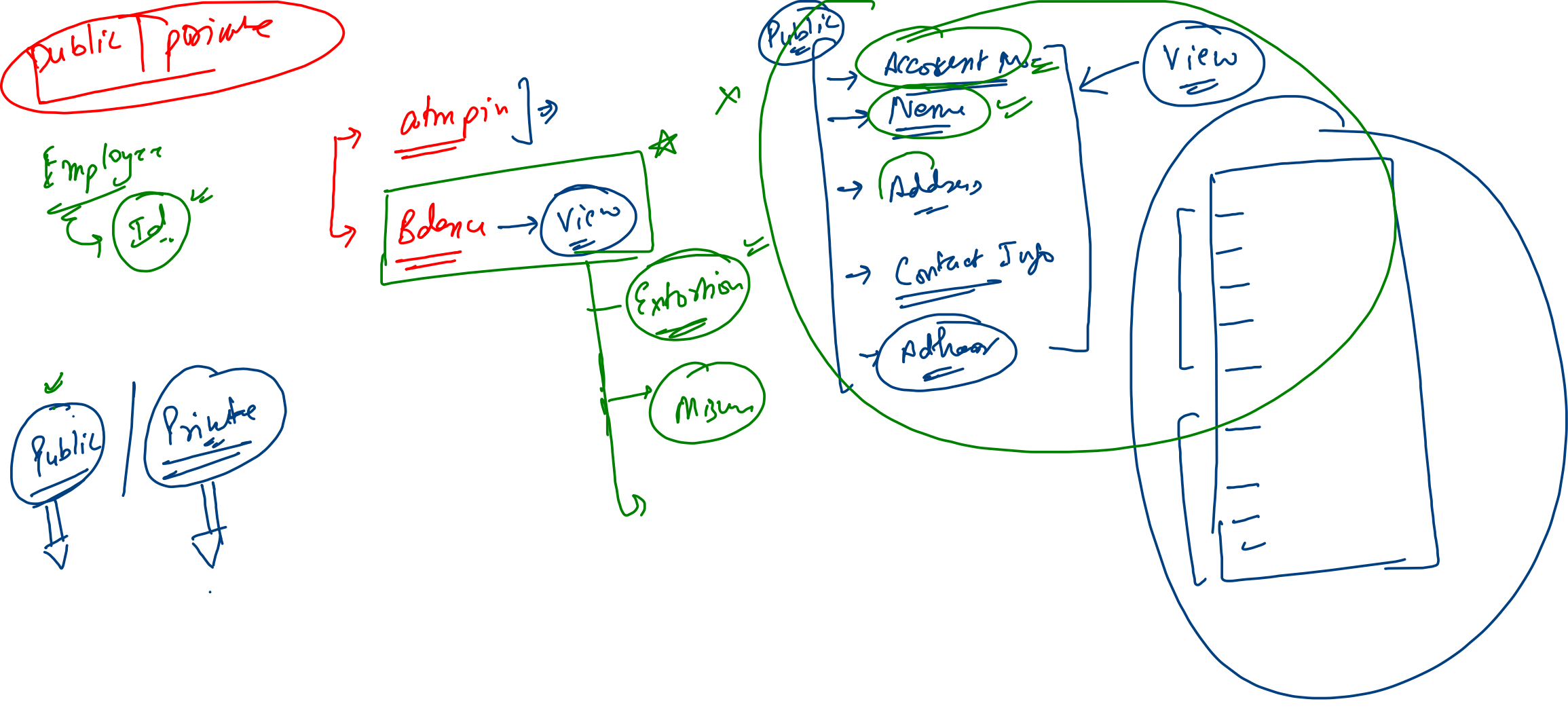
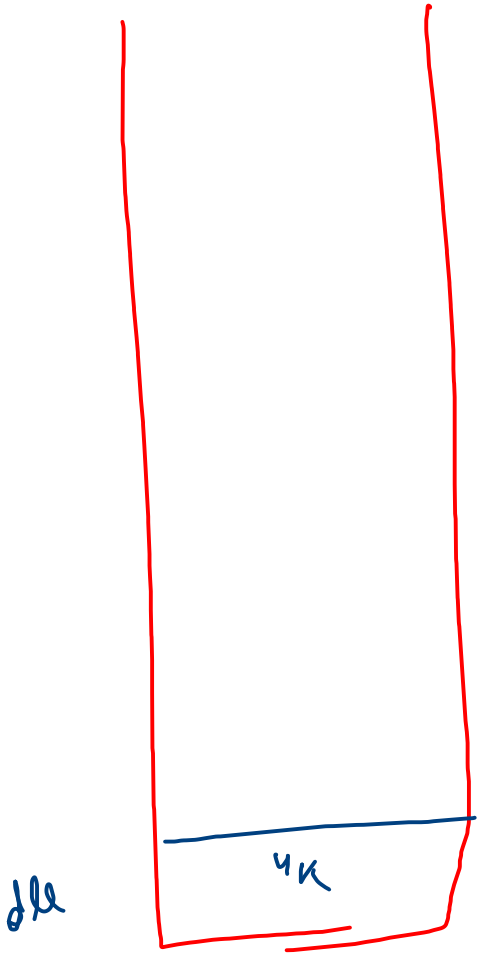

Level 1
Linked List
S.O.
Head
Tail
Size
Node
data
next
SLL
Level 2
Linked List
S.O.
H.
T
S
Node
data
next
prev
DLL

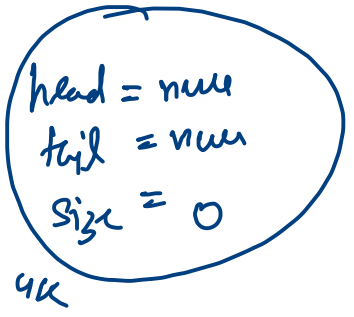
</>	Add First In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	28
</>	Add Last In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	29
</>	Remove First In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	30
</>	Remove Last In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	31
</>	Get First And Get Last In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	32
</>	Get At In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	33
</>	Add At In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	34
</>	Remove At In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	35
</>	Add Before In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	36
</>	Add After In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	37
</>	Remove After In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	38
</>	Remove Before In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	39
</>	Remove Node In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	40
</>	Display Forward And Backward In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	41


Public | private
atmpin
Employee
Id
Balance
View
Extension
Mpin
Public
Private
Public
Account No
Name
Address
Contact Info
Adhaar
View

Access Specifier

```java
public static class DoublyLinkedList {
  private class Node {
    int data = 0;
    Node prev = null;
    Node next = null;

    Node(int data) {
      this.data = data;
    }

  }

  private Node head = null;
  private Node tail = null;
  private int size = 0;

  public String toString() {⟺}

  public void addFirst(int val) {

  }

}

public static void main(String[] args) {
  Scanner scn = new Scanner(System.in);
  DoublyLinkedList dll = new DoublyLinkedList();
```





```
addFirst(int val) {
    Node node = new Node(val);
    if(size == 0) {
        Head = tail = node;
    }else {
        node.next = head;
        head.prev = node;
        head = node;
    }
    Size++;
}
```

head = 2K

tail = 5K

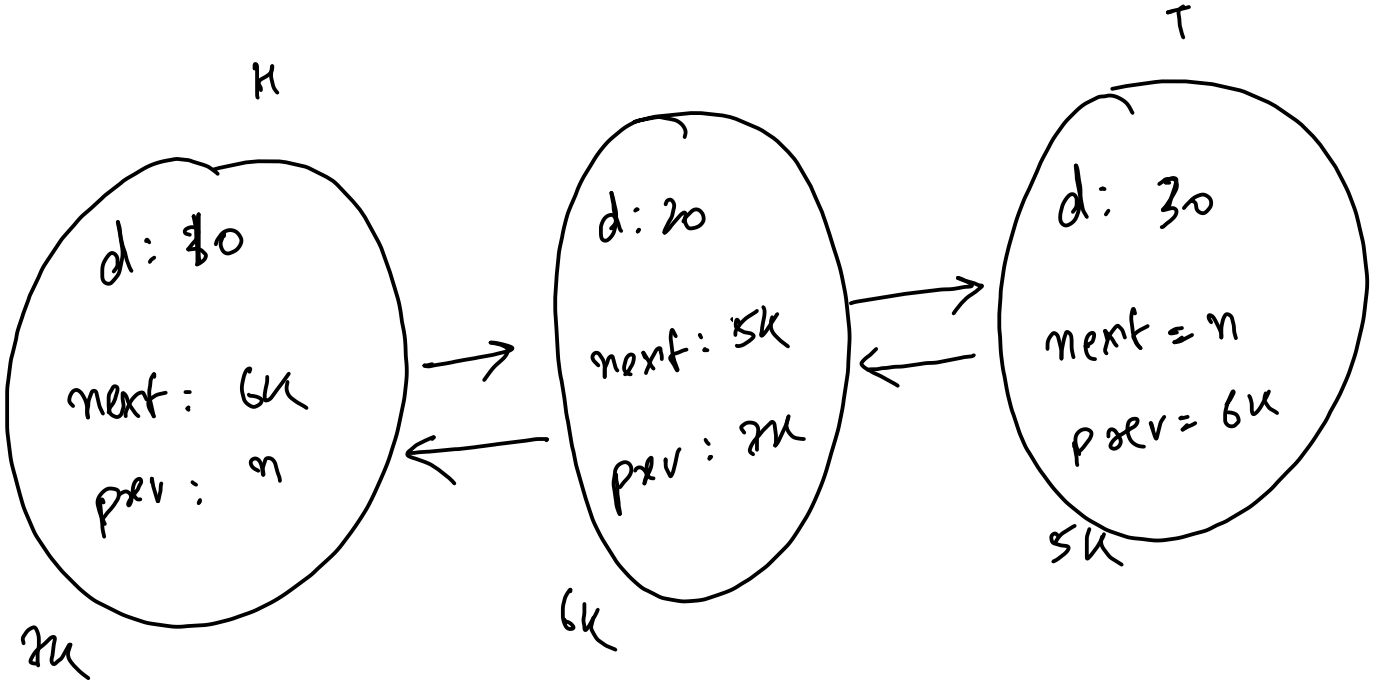
Size = 3

AddFirst

✓ add First (30)

✓ = (20)

✓ = (10)

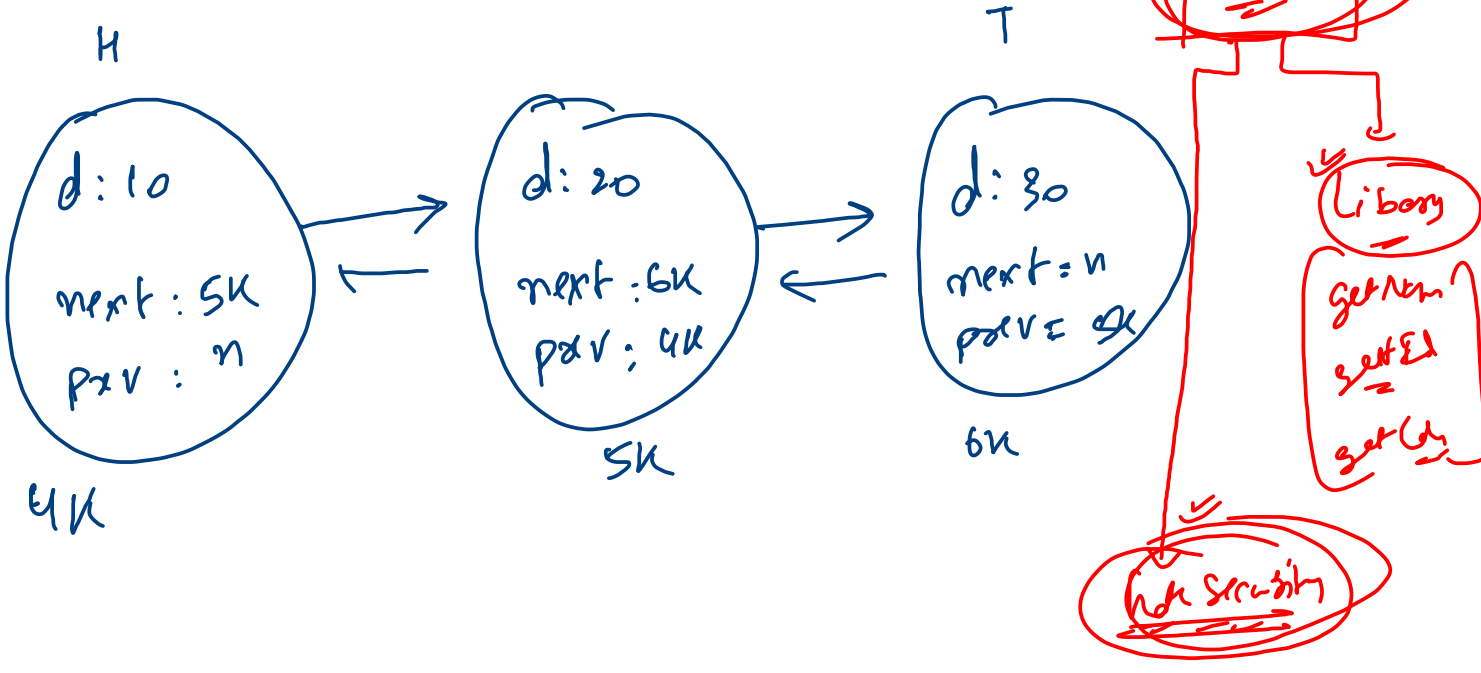


```
addLast(val) {
  Node node = new Node(val);

  if (size == 0) {
    head = tail = node;
  } else {
    tail.next = node;
    node.prev = tail;

    tail = node;
  }

  size++;
}
```

head : 4K
tail : 6K
size : 3

Add Last =

- ✓ addLast(10)
- ✓ = (20)
- ✓ = (30)



Remove First

head = null

tail = null

Size = 0

val = 40

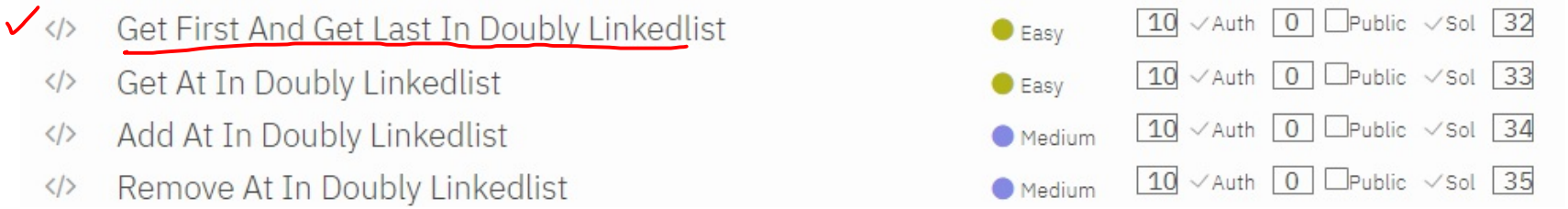
```
removeFirst() {
  if (size == 0) {
    print("List is empty")
  } else if (size == 1) {
    int val = head.data;
    head = tail = null
    size = 0;
    return val;
  } else {
    Node nbr = head.next;
    head.next = null;
    nbr.prev = null;
    int val = head.data;
    head = nbr
    size--;
    return val;
  }
}
```

Remove Link

head = 4K
tail = 4K
size = 1

NT
10
4K

val = 20


Get First And Get Last In Doubly Linkedlist
Easy
10 Auth 0 Public Sol 32
Get At In Doubly Linkedlist
Easy
10 Auth 0 Public Sol 33
Add At In Doubly Linkedlist
Medium
10 Auth 0 Public Sol 34
Remove At In Doubly Linkedlist
Medium
10 Auth 0 Public Sol 35

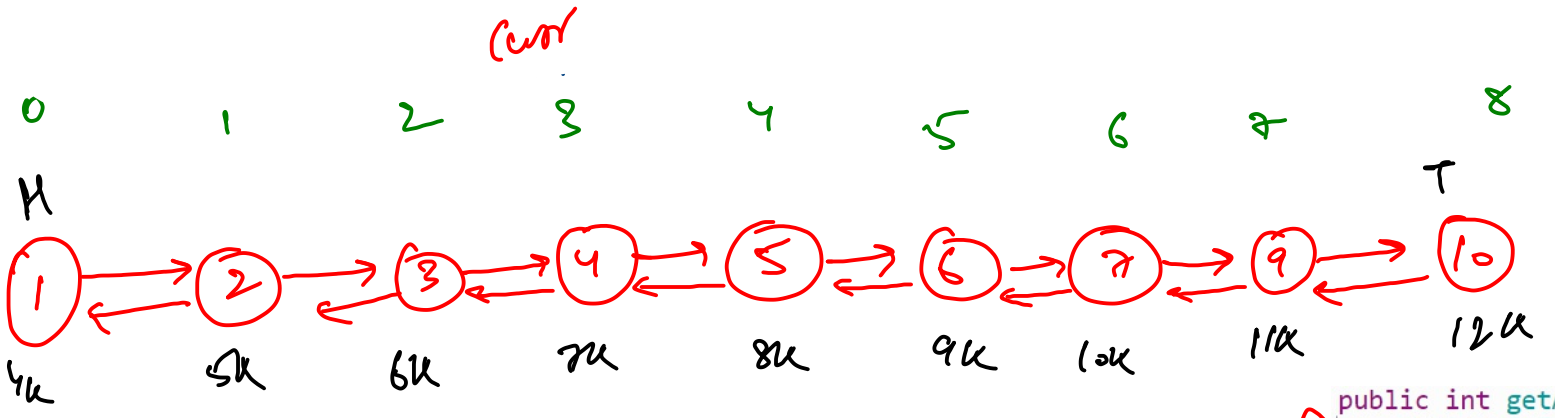
List is empty → size = 0

Invalid Index → idx < 0 || idx >= size-1

idx = 5

getAt(3)

index = 3 2 1 0




```
public int getAt(int index) {
    if(ListIsEmptyException()){
        return -1;
    }else if(indexIsInvalidException(index,0,size-1)){
        return -1;
    }else{
        Node curr = head;
        while(index != 0){
            curr = curr.next;
            index--;
        }
        return curr.data;
    }
}
```

4

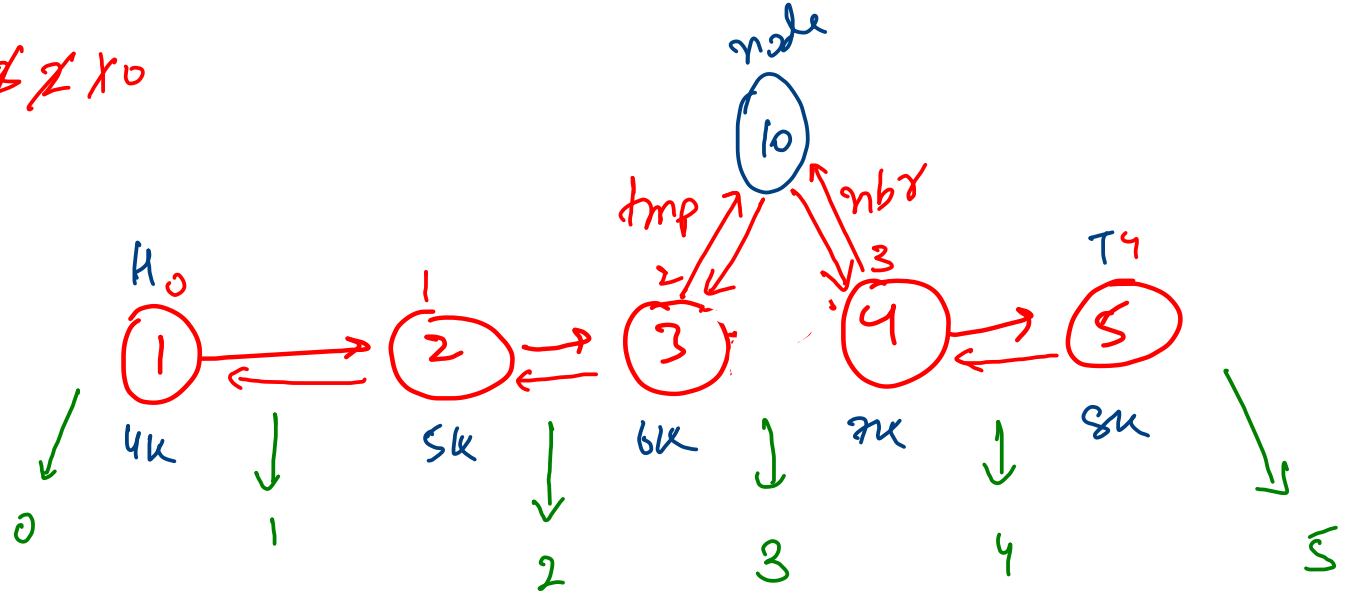
invalid index → idx < 0 || idx > size

idx == 0 → addFirst

idx == size → addLast

addAt(idx, val)

idx = 0 1 2 3 4 5




```
public void addAt(int index, int data) {
    if(indexIsInvalidException(index,0,size)){
        System.out.println("-1");
        return;
    }else{
        if(index == 0){
            addFirst(data);
        }else if(index == size){
            addLast(data);
        }else{
            index--;
            Node tmp = head;
            while(index != 0){
                tmp = tmp.next;
                index--;
            }

            Node nbr = tmp.next;
            Node node = new Node(data);
            tmp.next = node;
            node.prev = tmp;
            node.next = nbr;
            nbr.prev = node;

            size++;
        }
    }
}
```

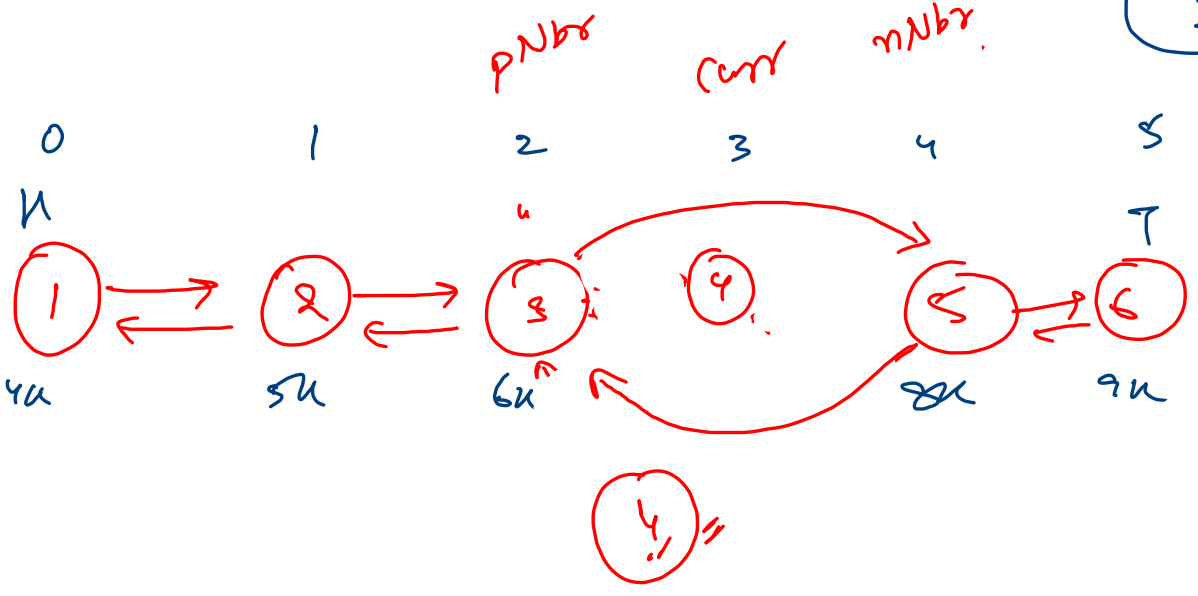
List IsEmpty → size == 0

Invalid Index → idx < 0 || idx > size-1

idx == size-1

removeLast

idx == 0

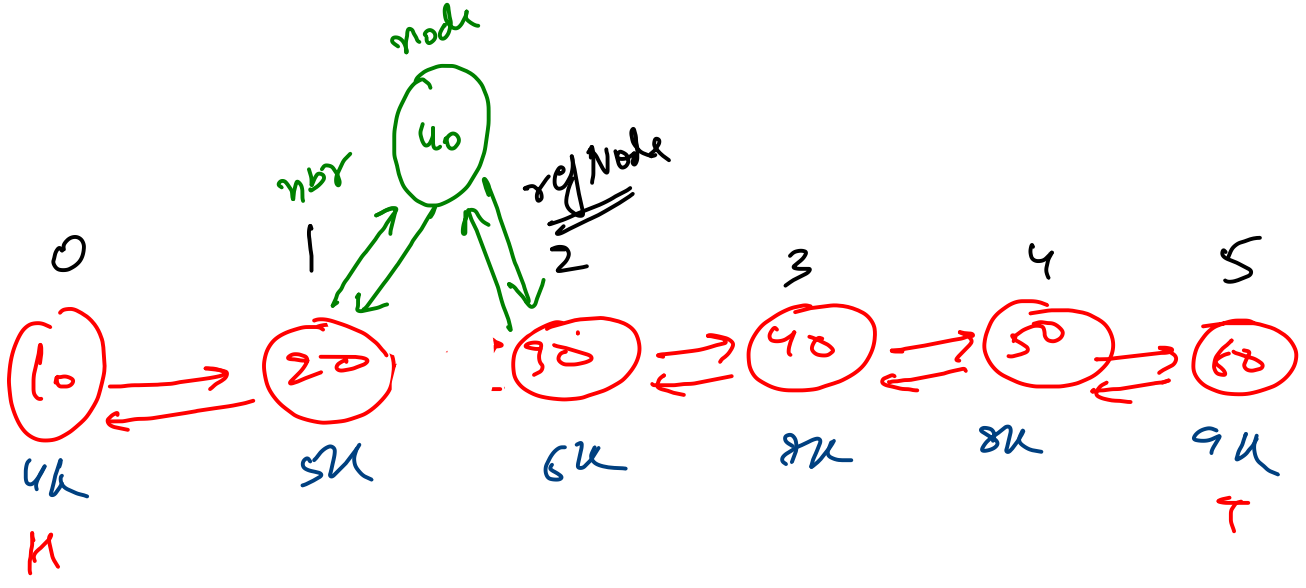



```
public int removeAt(int index) {
    if(ListIsEmptyException()){
        return -1;
    }else if(indexIsInvalidException(index,0,size-1)){
        return -1;
    }else{
        if(index == 0){
            return removeFirst();
        }else if(index == size-1){
            return removeLast();
        }else{
            Node curr = head;

            while(index != 0){
                curr = curr.next;
                index--;
            }

            Node pnbr = curr.prev;
            Node nnbr = curr.next;
            pnbr.next = nnbr;
            nnbr.prev = pnbr;
            curr.next = null;
            curr.prev = null;

            size--;
            return curr.data;
        }
    }
```

aB(2, 40)

```
public void addBefore(Node refNode, int data) {
  // complete this function.
}

public void addBefore(int idx, int data) {
  Node node = getNodeAt(idx);
  addBefore(node, data);
}
```

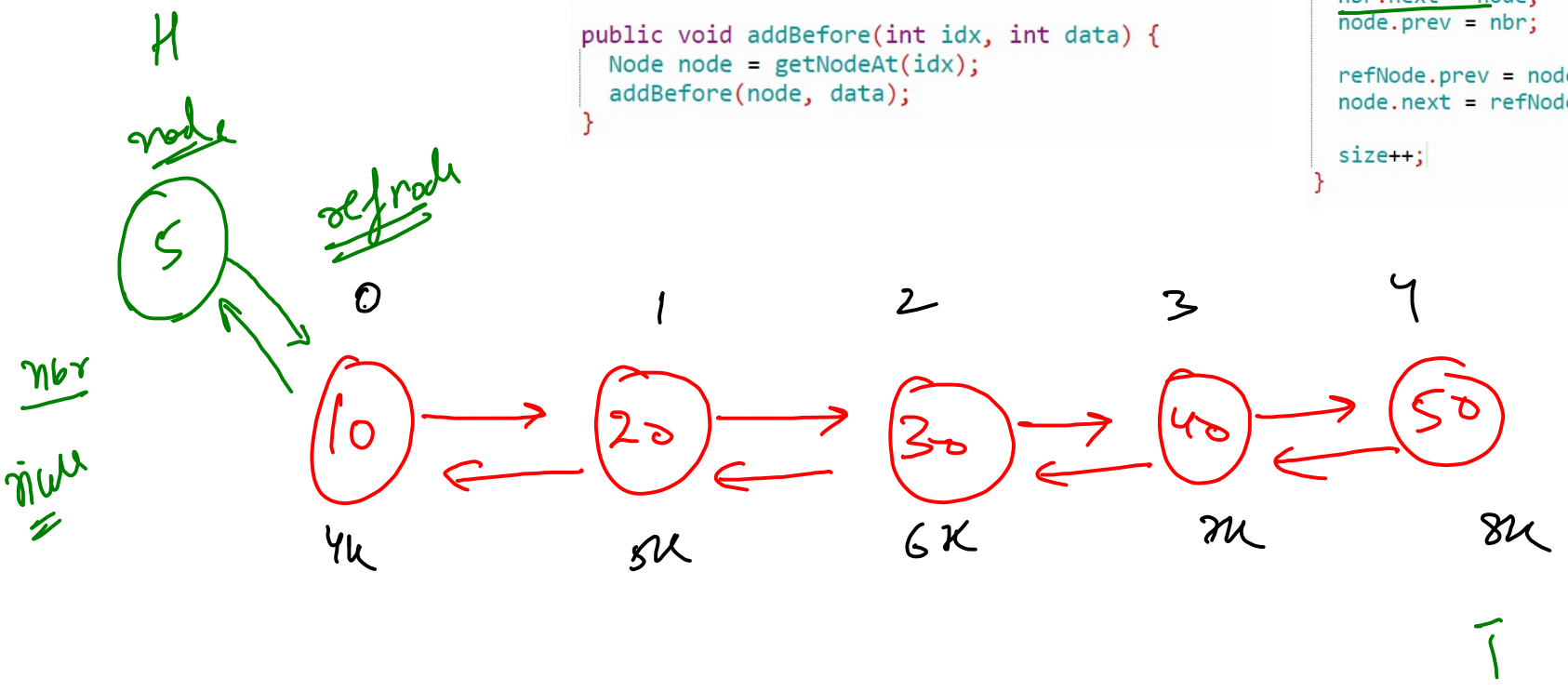
```
public void addBefore(Node refNode, int data) {
  Node node = new Node(data);
  Node nbr = refNode.prev;

  nbr.next = node;
  node.prev = nbr;

  refNode.prev = node;
  node.next = refNode;

  size++;
}
```



addBefore(0, 5)

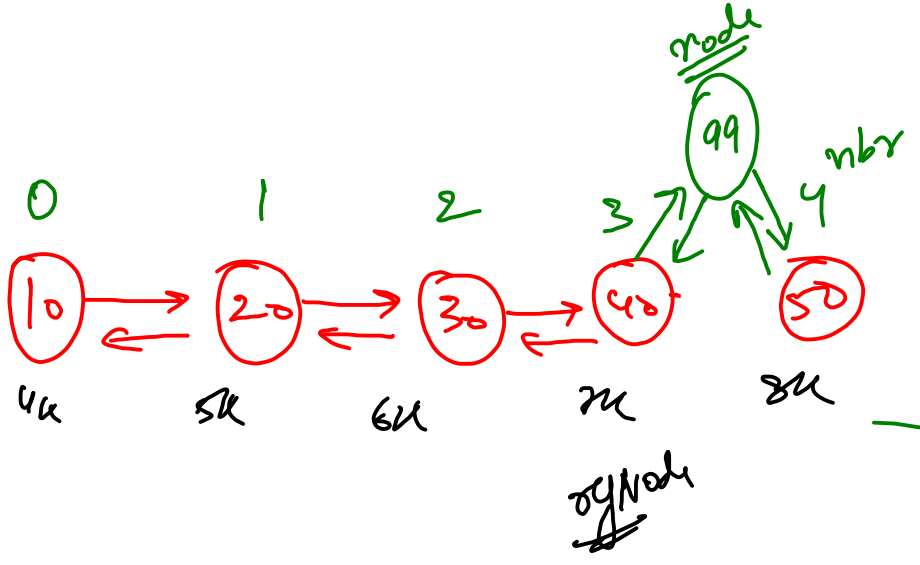
```
public void addBefore(Node refNode, int data) {
  // complete this function.
}

public void addBefore(int idx, int data) {
  Node node = getNodeAt(idx);
  addBefore(node, data);
}
```

```
public void addBefore(Node refNode, int data) {
  Node node = new Node(data);
  Node nbr = refNode.prev;

  nbr.next = node;
  node.prev = nbr;

  refNode.prev = node;
  node.next = refNode;

  size++;
}
```



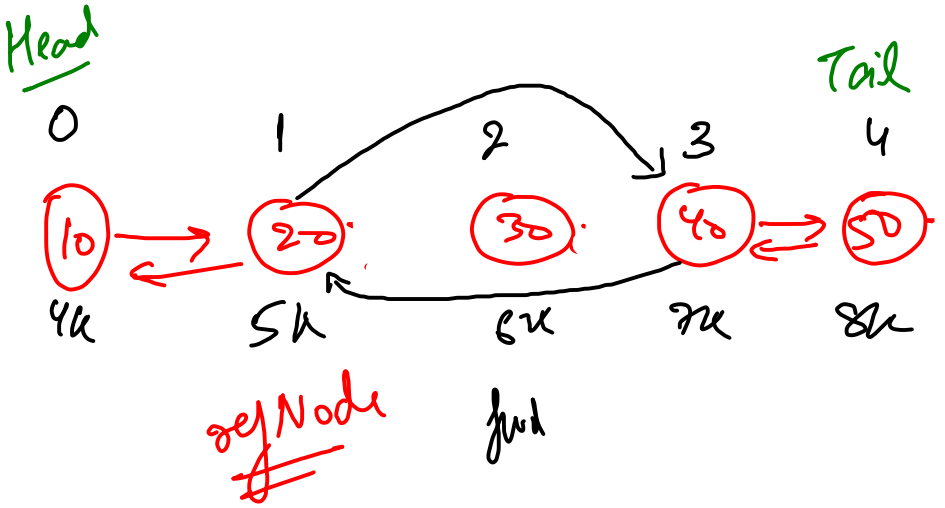
```java
public void addAfter(Node refNode, int data) {

}

public void addAfter(int idx, int data) {
  Node node = getNodeAt(idx);
  addAfter(node, data);
}
```

addAfter(3, 99)

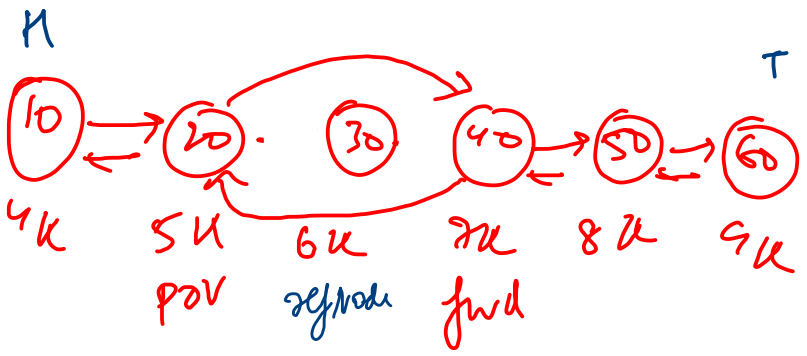


```java
public void addAfter(Node refNode, int data) {
    Node node = new Node(data);
    Node nbr = refNode.next;

    if(nbr == null){
        refNode.next = node;
        node.prev = refNode;
        tail = node;
    }else{
        node.next = nbr;
        nbr.prev = node;

        refNode.next = nbr;
        nbr.prev = refNode;
    }

    size++;
}
```

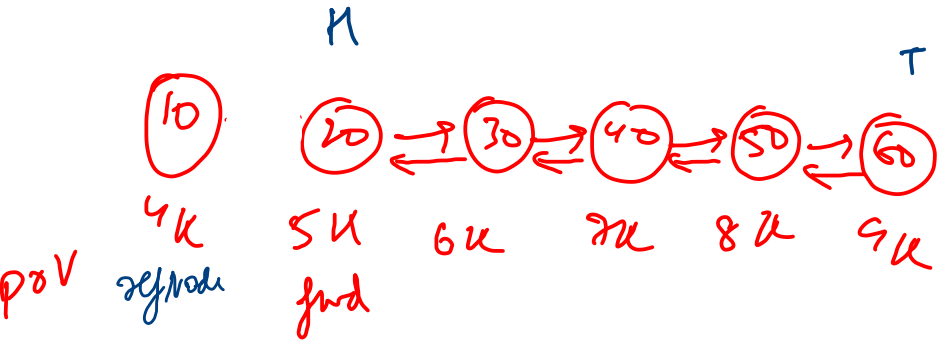
removeAfter(1)

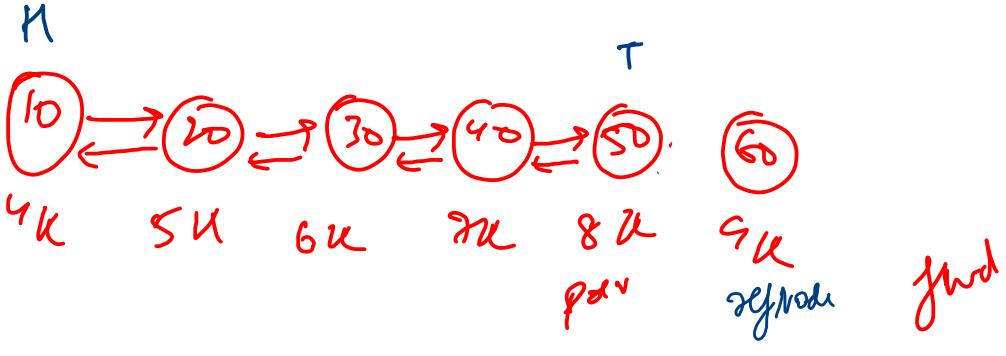



```java
public int removeAfter(Node refNode) {
    if(refNode.next == null){
        System.out.println("LocationIsInvalid: ");
        return -1;
    }else{
        Node fwd = refNode.next;

        if(fwd.next == null){
            fwd.prev = null;
            refNode.next = null;
            tail = refNode;
        }else{
            refNode.next = fwd.next;
            fwd.next.prev = refNode;
            fwd.next = null;
            fwd.prev = null;
        }

        size--;
        return fwd.data;
    }
}
```


H
T
10
20
30
40
50
60
4к
5к
6к
7к
8к
9к
prev
удалить
fwd


H
T
num
10
num
prev
удалить
next


H
T
10
20
30
40
50
60
4к
5к
6к
7к
8к
9к
prev
удалить
fwd


H
T
10
20
30
40
50
60
4к
5к
6к
7к
8к
9к
prev
удалить
fwd

</>	Add First In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	28
</>	Add Last In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	29
</>	Remove First In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	30
</>	Remove Last In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	31
</>	Get First And Get Last In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	32
</>	Get At In Doubly Linkedlist	Easy	10	Auth	0	Public	Sol	33
</>	Add At In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	34
</>	Remove At In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	35
</>	Add Before In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	36
</>	Add After In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	37
</>	Remove After In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	38
</>	Remove Before In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	39
</>	Remove Node In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	40
</>	Display Forward And Backward In Doubly Linkedlist	Medium	10	Auth	0	Public	Sol	41