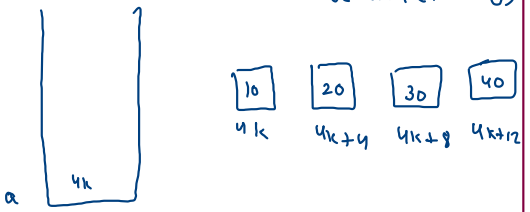
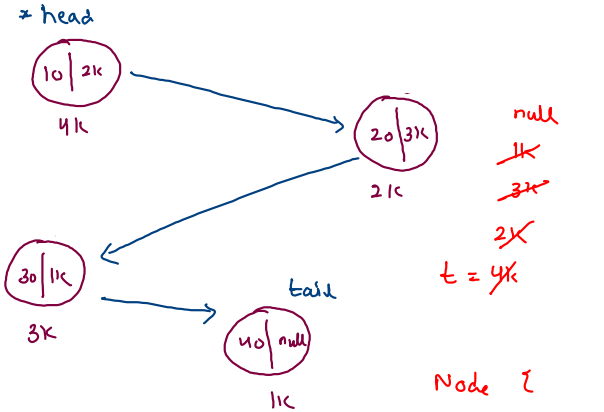
Array

```
int [] a = new int [4];
```

Continous memory allocation (indexing)



Linked List

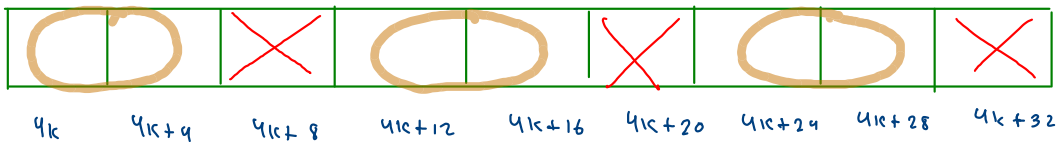



```
Node {
    int data;
    Node next;
}
```

10 20 30 40

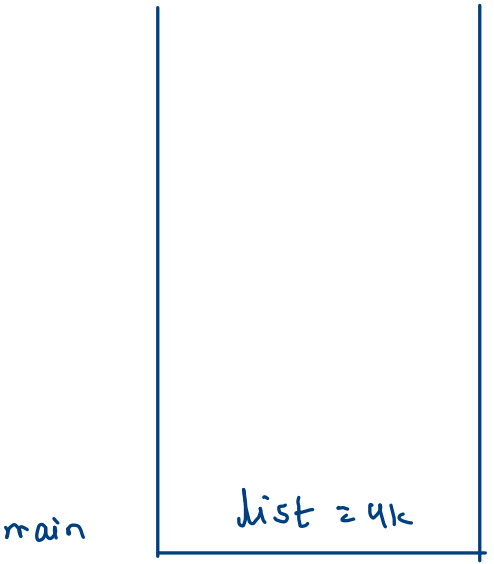
3 size array -> faild (no continous 3 free spots are there)

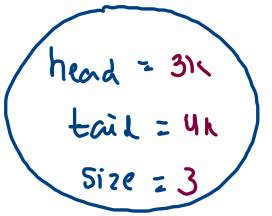
3 size LL -> possible (random memory allocation)

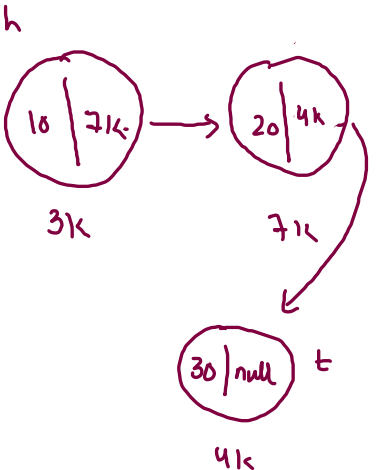


```java
public static class Node {
  int data;
  Node next;
}

public static class LinkedList {
  Node head;
  Node tail;
  int size;

  void addLast(int val) {
    // Write your code here
  }
}
```

```java
LinkedList list = new LinkedList();
```







✓ list.addLast(10);

✓ list.addLast(20);

✓ list.addLast(30);

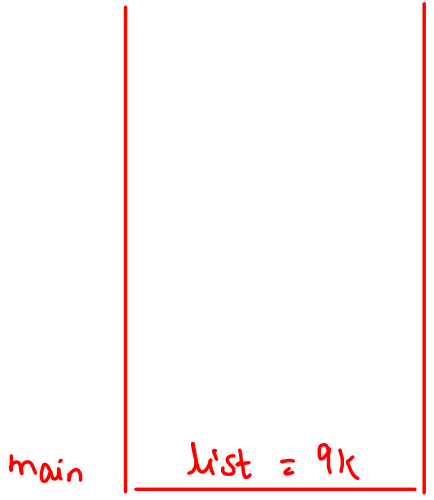
```java
public static class Node {
  int data;
  Node next;
}

public static class LinkedList {
  Node head;
  Node tail;
  int size;

  void addLast(int val) {
    Node nn = new Node();
    nn.data = val;

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

    size++;
  }
}
```

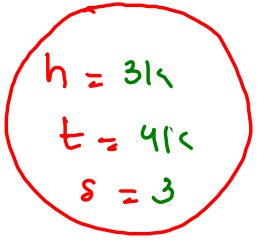
```java
LinkedList list = new LinkedList();
```

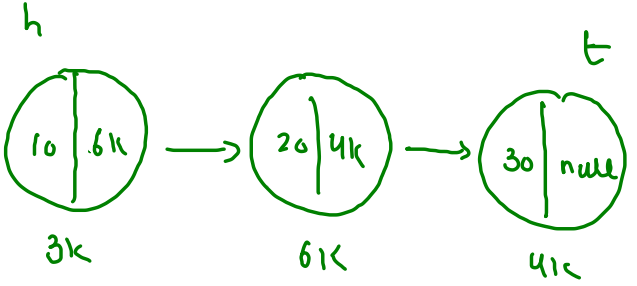
✓ list.al(10);

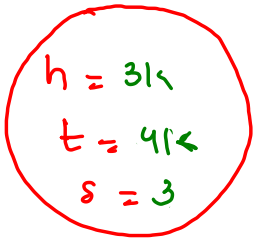
✓ list.al(20);

✓ list.al(30);

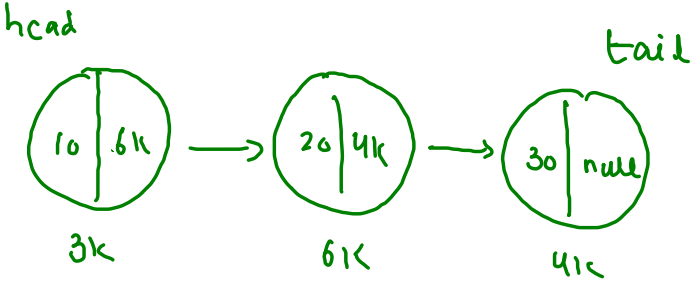








9k (LL)



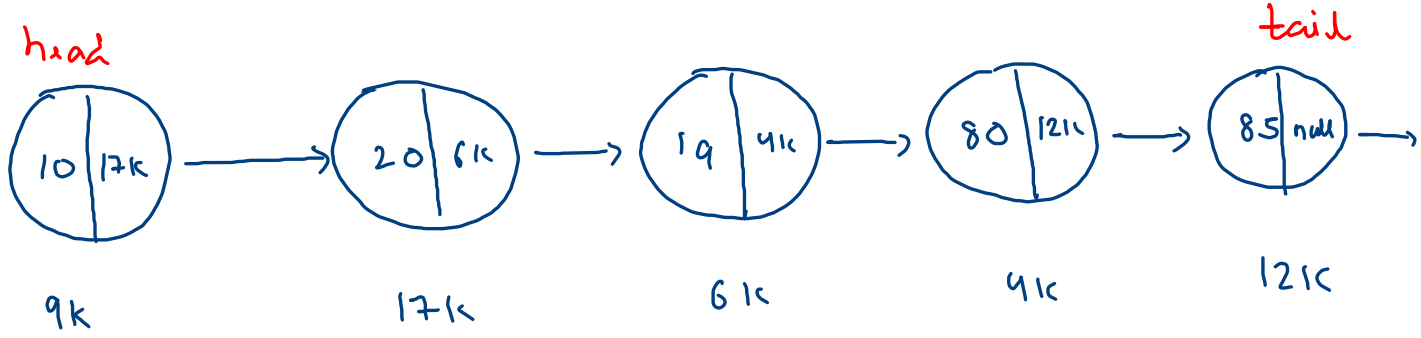
temp = null

```
list.display();

temp = head;
while (temp != null) {
    syso (temp.data);
    temp = temp.next;
}
```

10 20 30

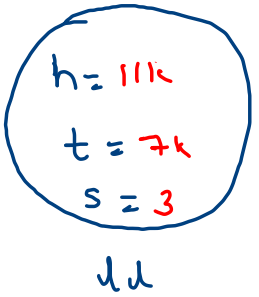
```java
public void display(){
    Node temp = head;

    while(temp != null) {
        System.out.print(temp.data + " ");
        temp = temp.next;
    }

    System.out.println();
}
```

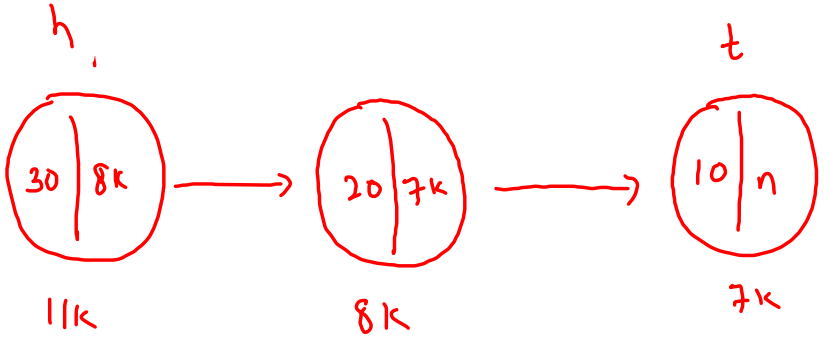
head = 9k
tail = 12k
S = S



10 20 19 80 85

Add First In Linked List





nn.next = head

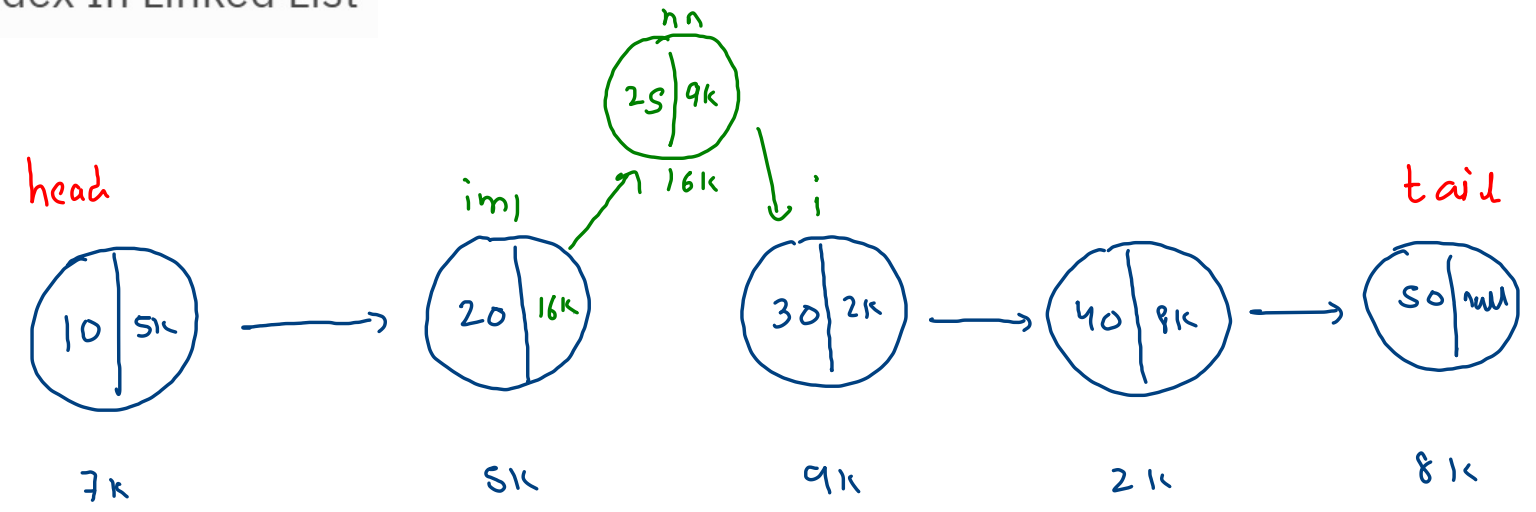
head = nn

ll.aF(10)

ll.aF(20)

ll.aF(30)

Add At Index In Linked List

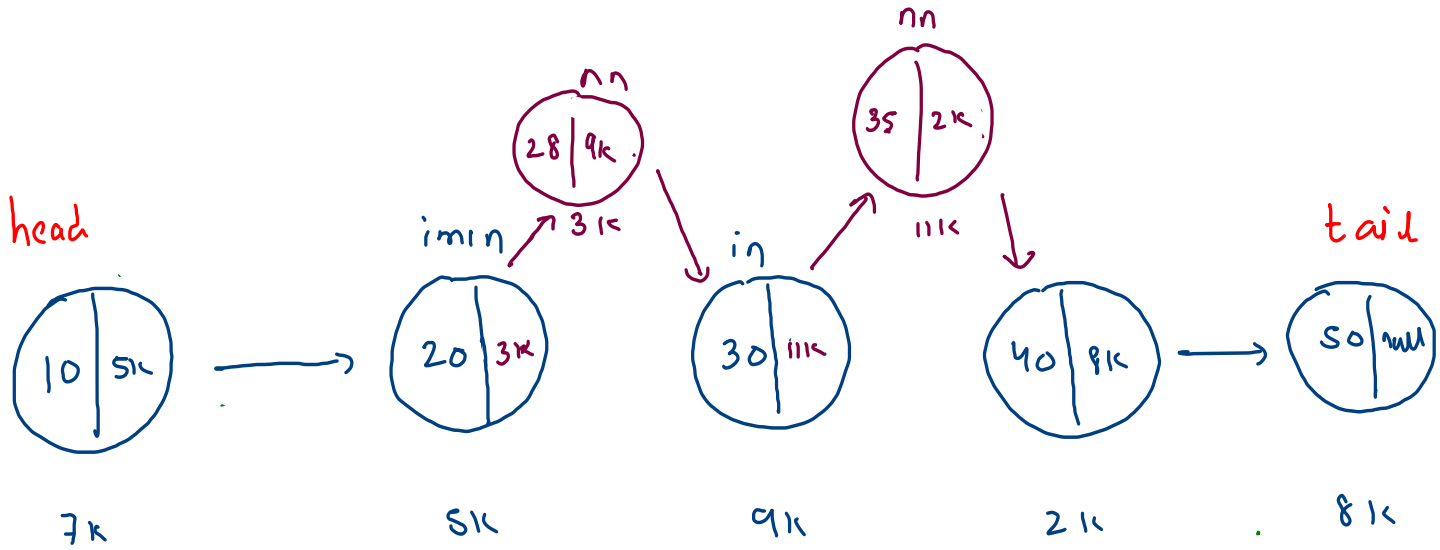


M. addAt (2 , 25)

valid posn : 0 to size

(i) 0 → aF (ii) size → aL

(iii) 1 to size-1

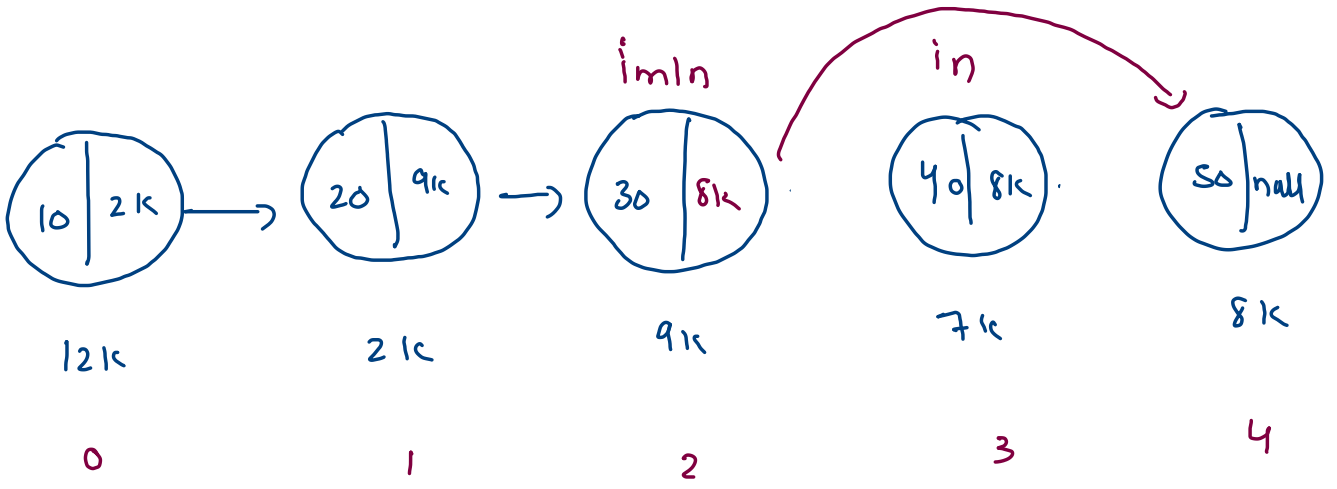



```java
public void addAt(int idx, int val){
    if(idx < 0 || idx > size) {
        System.out.println("Invalid arguments");
    }
    else if(idx == 0) {
        addFirst(val);
    }
    else if(idx == size) {
        addLast(val);
    }
    else {
        Node im1n = getNodeAt(idx-1); //(idx-1)'s node
        Node in = im1n.next; //idx node

        //insert new node between im1n and in
        Node nn = new Node();
        nn.data = val;

        im1n.next = nn;
        nn.next = in;

        size++;
    }
```

addAt(3,35);

addAt(2,28);

Remove At Index In Linked List



imIn.next = in.next

in.next = null;

removeAt(3);