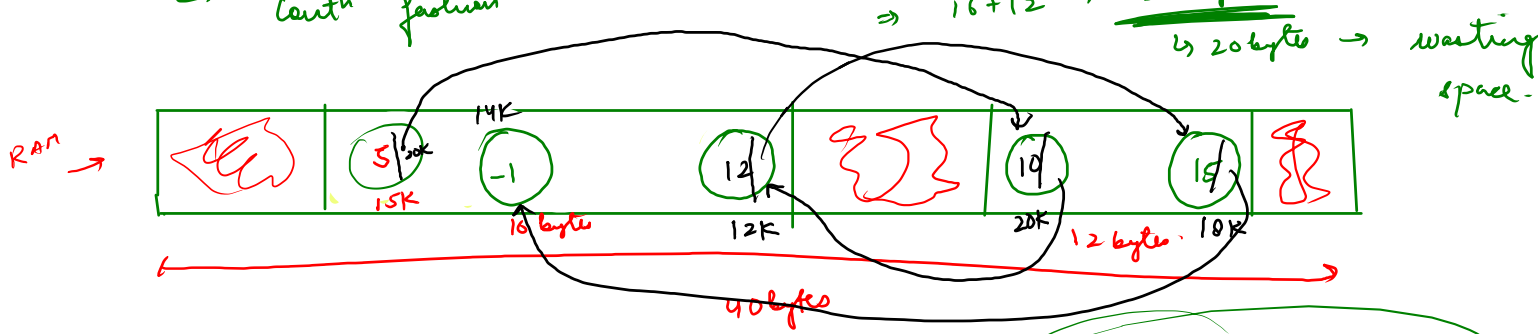
Data Structure → Linked List

↳ Array

↳ size is always fixed.

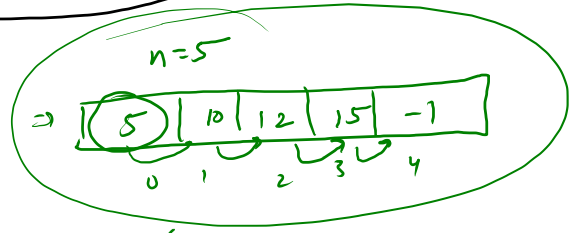
↳ contn fashion



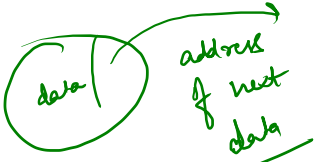
⇒ 5 int → 4×5 = 20 bytes

↳ in contn fashion

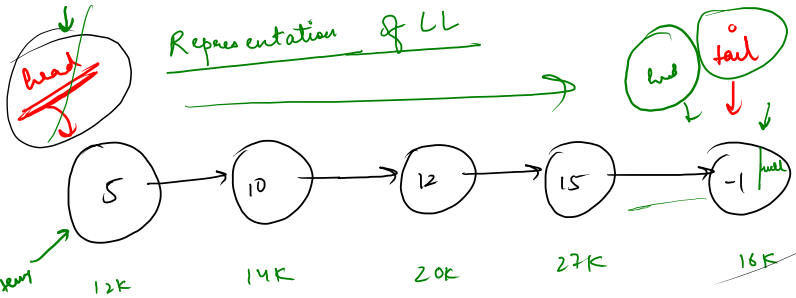
int → 4 byte.



⇒ Linked list D.S.



→ node.



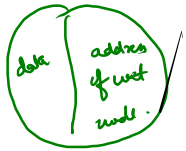
⇒ first node = head

⇒ last node = tail

⇒ 2 mins

Note:- Never ever change the position of head pointer.

Structure of Node




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

Structure of LL

head: 15K

tail: 16K

size: 5

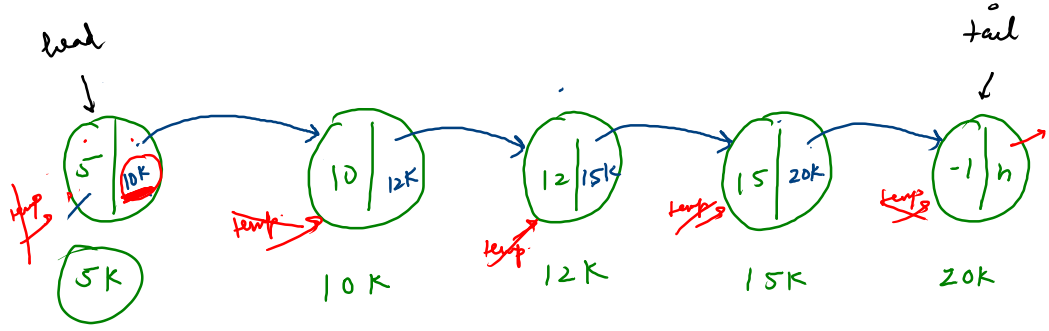
```java
public static class Node {
    int data;
    Node next;

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

```java
public static void display(Node head) {
    if (head == null) return;

    Node temp = head;
    while(temp != null) {
        System.out.println(temp.data);
        temp = temp.next;
    }
}
```

10 mins → Code + Dry Run

head == null → no node is present in a LL

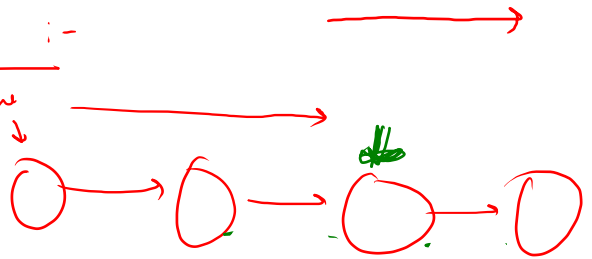
```java
public static void main(String[] args) {
    /* Enter your code here. Read input fr

    Node a = new Node(5);
    Node b = new Node(10);
    Node c = new Node(12);
    Node d = new Node(15);
    Node e = new Node(-1);

    a.next = b;
    b.next = c;
    c.next = d;
    d.next = e;

    Node head = a;
    Node tail = e;

    display(head);
}
```



temp = 5K 10K 12K 15K 20K null

5
10
12
15
-1

Singly LL :-

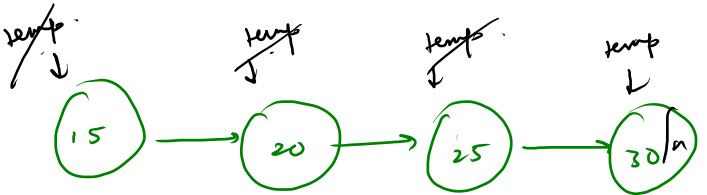


head == null ⇒ no node is present in the LL

head.next == null ⇒ only one node is present in LL

temp = head; while (temp != null) ⇒ traversing all the nodes of a LL

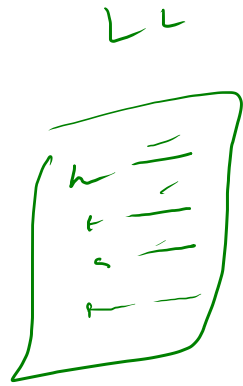
temp = head; while (temp.next != null) ⇒ traversing till the last node.



Add Last In Linked List

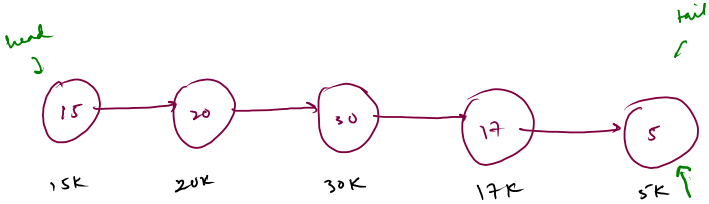
Problem | Submissions | Leaderboard | Discussions

1. You are given a partially written LinkedList class.
2. You are required to complete the body of addLast function. This function is supposed to add an element to the end of LinkedList. You are required to update head, tail and size as required.
3. Input and Output is managed for you. Just update the code in addLast function.



Add last in a LL (data) = 12

→ head = = null.



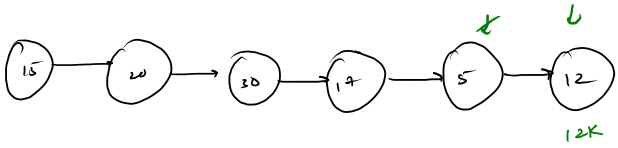
head → 15K
tail → 5K
size → 5

return

head
tail
size –

I/P

O/P



head → 15K
tail → 12K
size → 6

→ → a

tail . next = a;

tail = a
size++;

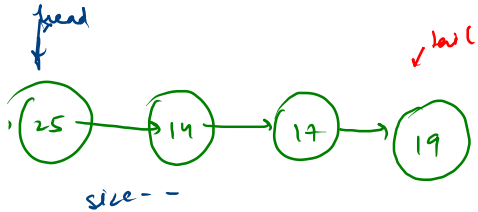
temp = head
while (temp.next != null) → temp = temp.next;
temp = temp.next;

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

    if(size == 0) {
        tail = n;
        head = n;
    }  else {
        tail.next = n;
        tail = n;
    }
      size++;
  }
}
```

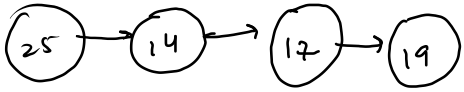
Remove first in a LL :-

3. You are required to complete the body of removeFirst function 3.1. removeFirst - This function is required to remove the first element from Linked List. Also, if there is only one element, this should set head and tail to null. If there are no elements, this should print "List is empty".

I.P.



O.P.



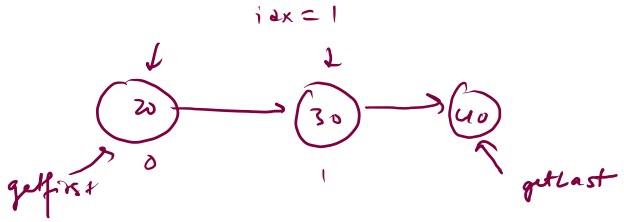

```
if(size==1)
{
    head=null
    tail=null
    size=0
}
```

if(size==0)
↓
size==0
(head==null)

↳ List is empty

```java
public void removeFirst() {
    if (size == 0) {
        System.out.println("List is empty");
        return;
    } else if (size  == 1) {
        head = null;
        tail = null;
        size = 0;
    } else {
        head = head.next;
        size--;
    }
}
```

Get value in LL :-



3

idx = 10

idx = -1

idx = 2

3.1. getFirst - Should return the data of first element. If empty should return -1 and print "List is empty".

3.2. getLast - Should return the data of last element. If empty should return -1 and print "List is empty".

3.3. getAt - Should return the data of element available at the index passed. If empty should return -1 and print "List is empty". If invalid index is passed, should return -1 and print "Invalid arguments".

idx

→ getFirst
- size == 0 → return -1 → List is empty.
- size != 0 → return head.data;

→ getLast
- size == 0 → return -1, List is empty.
- size != 0 → return tail.data;

getAt (idx)

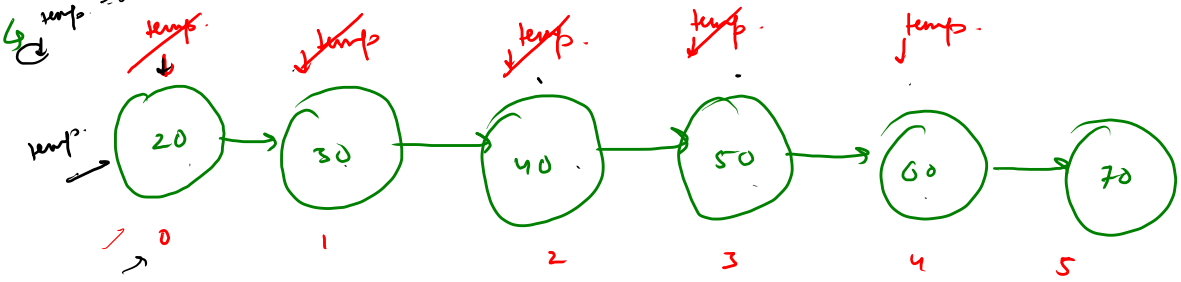
↳ (size == 0) → return -1, List is empty.

↳ if (idx < 0 || idx >= size) → return -1, "Invalid argument";

for(int i=0; i<4; i++)

idx = 4

↳ temp = head



for(int i=0; i<idx; i++)

temp = temp.next;

→ i = 0 < 4

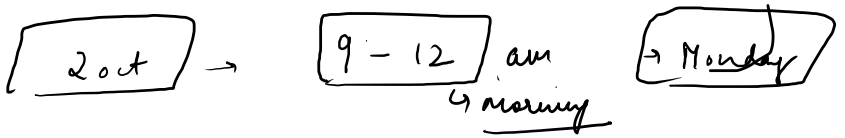
→ i = 1 < 4

→ i = 2 < 4

→ i = 3 < 4

i = 4 < 4

return temp.data;




```java
public int getFirst() {
    if (size == 0) {
        System.out.println("List is empty");
        return -1;
    }

    return head.data;
}

public int getLast(){
  if (size == 0) {
        System.out.println("List is empty");
        return -1;
    }

    return tail.data;
}
```

```java
public int getAt(int idx){
  if (size == 0) {
        System.out.println("List is empty");
        return -1;
    } else if (idx < 0 || idx >= size) {
      System.out.println("Invalid arguments");
      return -1;
  }  else {
      Node temp = head;
      for(int i=0;i<idx;i++) {
          temp = temp.next;
      }

      return temp.data;
  }
}
```