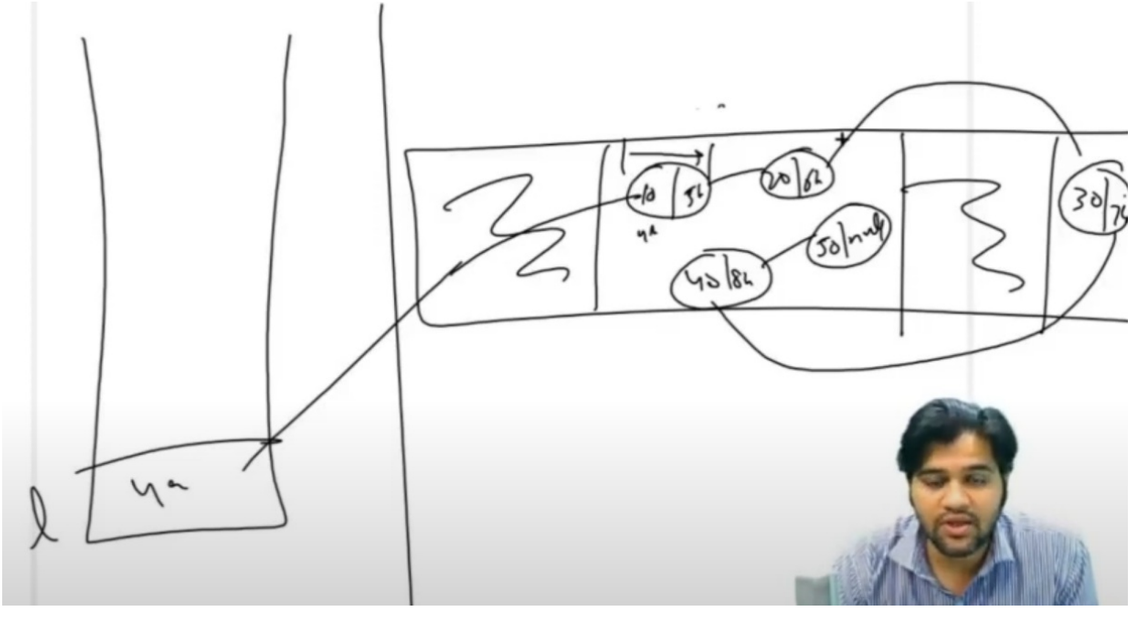
LinkedList

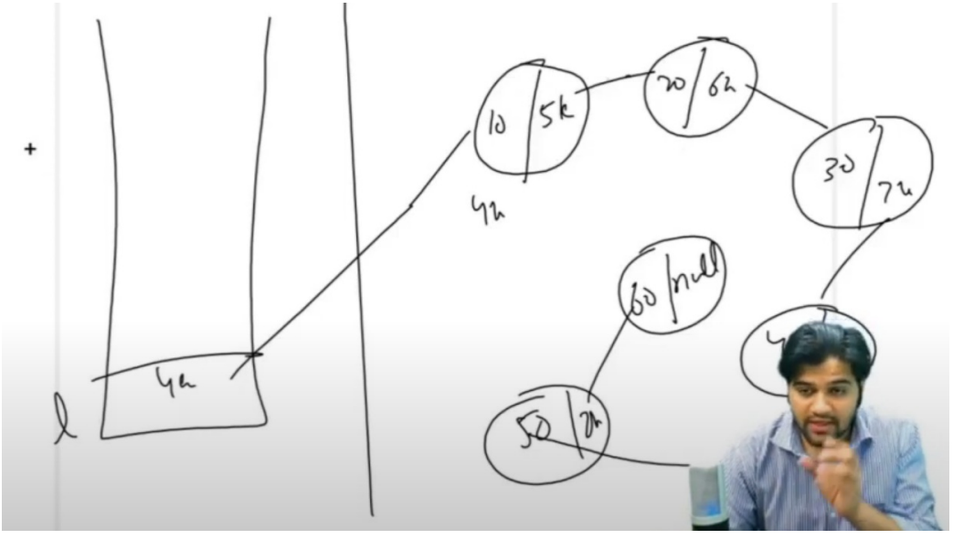
INTRO

array me continuous data store hota he (4byte int data)

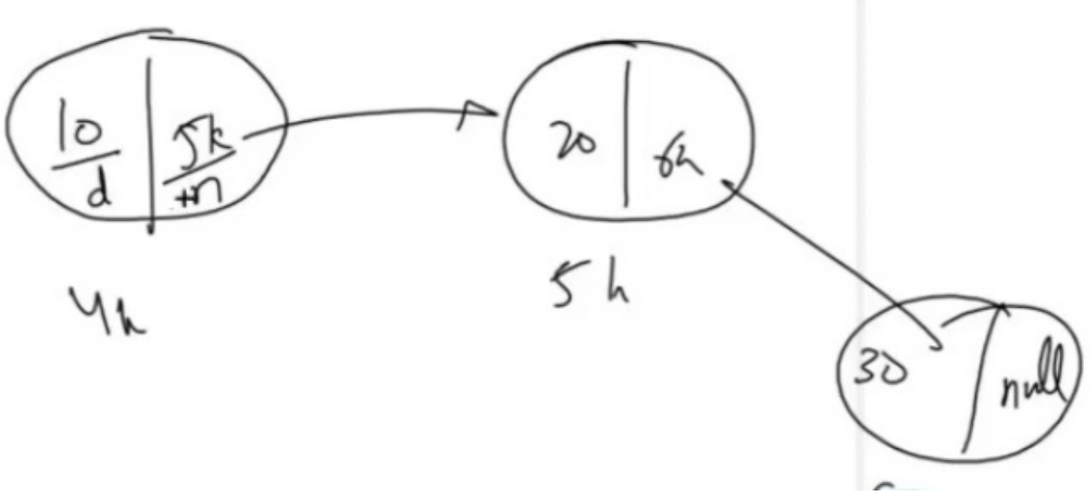
linklist me continuous data store nhi hota he

it has data and next address
(4byte int and 4byte address)
and last elemnet have address null



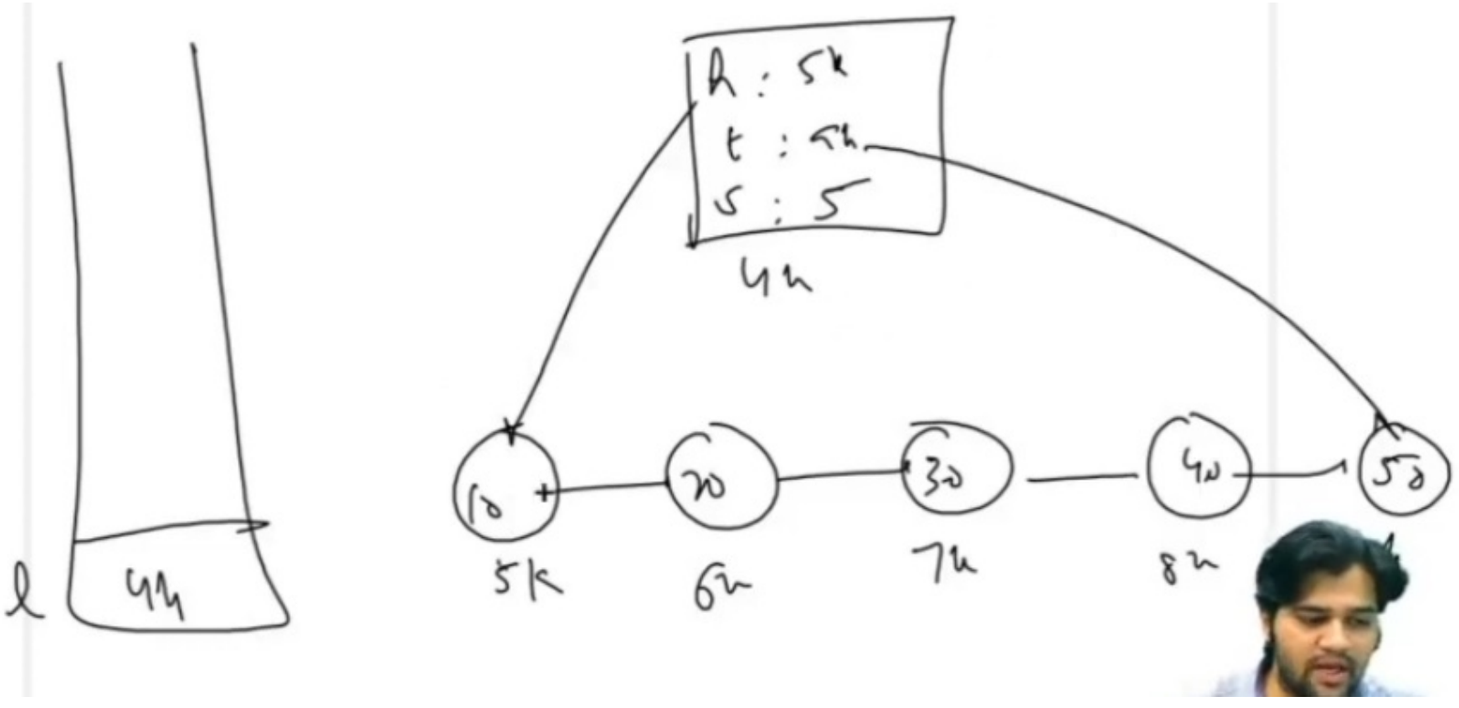


Data Members of a Linked List




```
2
3  public class Main {
4      public static class Node {
5          int data;
6          Node next;
7      }
8
9      public static void main(String[] args) {
10
11     }
12
```

Linklist




```
2
3  public class Main {
4      public static class Node {
5          int data;
6          Node next;
7      }
8
9      public static class LinkedList {
10         Node head;
11         Node tail;
12         int size;
13
14
15     }
16
17     public static void main(String[] args) {
18
19     }
20
21 }
```

head- 1st node
tail- last node
size- total nodes present

Add Last In Linked List

Easy

Prev Next

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.

Input Format

Input is managed for you

Output Format

Constraints

None

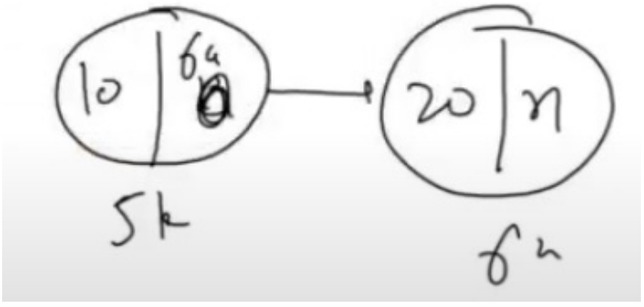
Sample Inpu

```
addLast 10
addLast 20
addLast 30
addLast 40
addLast 50
quit
```

Sample Outp

```
10
20
30
40
50
5
50
```

Steps to add node at last



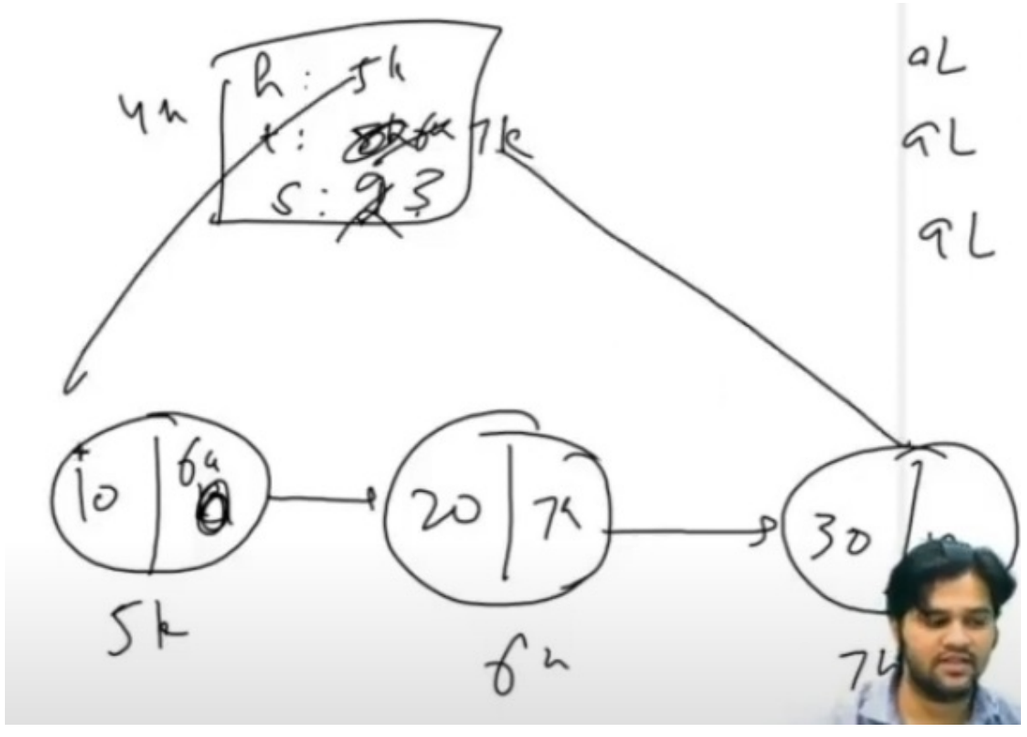
1.make new node
2. node->data = val
3. node->next -null
4. goto last node of linklist and update the node->next with address of new node
5.set current tail->next of linklist as address of above newnode
6. update size of linklist

For 1st node

```
3
4  public class Main {
5    public static class Node {
6      int data;
7      Node next;
8    }
9
10   public static class LinkedList {
11     Node head;
12     Node tail;
13     int size;
14
15     void addLast(int val) {
16       // Write your code here
17       if(size == 0){
18           Node temp = new Node();
19           temp.data = val;
20           temp.next = null;
21           head = tail = temp;
22           size++;
23       }
24     }
25   }
```

for other node, else part

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

  void addLast(int val) {
    // Write your code here
    if(size == 0){
        Node temp = new Node();
        temp.data = val;
        temp.next = null;

        head = tail = temp;

        size++;
    } else {
        Node temp = new Node();
        temp.data = val;
        temp.next = null;

        tail.next = temp;
        tail = temp;

        size++;
    }
  }
}
```



Clean node

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

  void addLast(int val) {
    Node temp = new Node();
    temp.data = val;
    temp.next = null;

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

    size++;
  }
}
```

Display A Linkedlist

Easy

Prev Next

1. You are given a partially written LinkedList class.
2. Here is a list of existing functions:
   2.1 addLast - adds a new element with given value to the end of Linked List
3. You are required to complete the body of display function and size function
   3.1. display - Should print the elements of linked list from front to end in a single line. Elements should be separated by space.
   3.2. size - Should return the number of elements in the linked list.
4. Input and Output is managed for you.

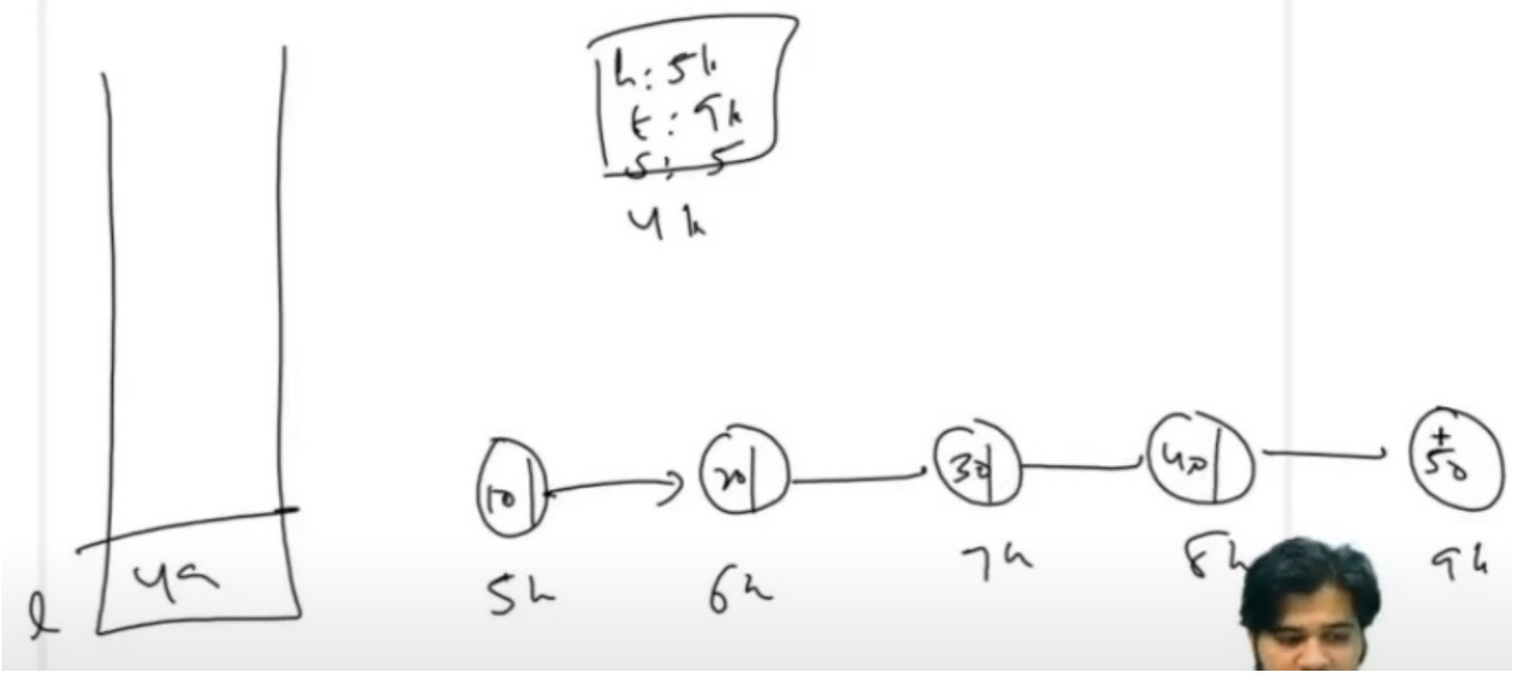
Input Format

Sample Input

```
addLast 10
addLast 20
addLast 30
display
size
addLast 40
addLast 50
```

Sample Output

```
10 20 30
3
10 20 30 40 50
5
```




```
public void display(){
    // write code here
    Node temp = head;
    while(temp != null){
        System.out.print(temp.data + " ");
        temp = temp.next;
    }
    System.out.println();
}
```

Remove First In Linkedlist

Easy

Prev Next

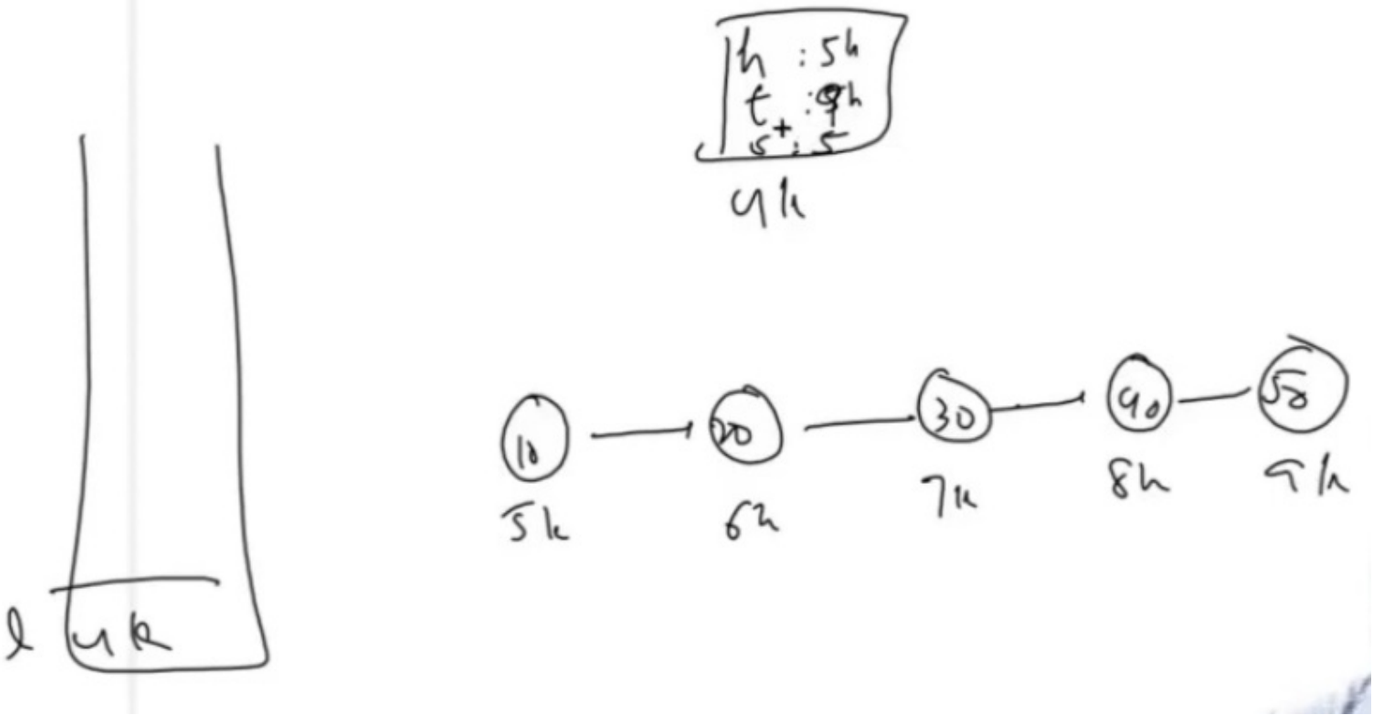
1. You are given a partially written LinkedList class.
2. Here is a list of existing functions:
   2.1 addLast - adds a new element with given value to the end of Linked List
   2.2. display - Prints the elements of linked list from front to end in a single line. All elements are separated by space
   2.3. size - Returns the number of elements in the linked list.
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".
4. Input and Output is managed for you.

Sample Input

```
addLast 10
addLast 20
addLast 30
display
removeFirst
size
addLast 40
```

Sample Output

```
10 20 30
2
30 40 50
3
List is empty
```




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

Get Value In Linked List

Easy

Prev

1. You are given a partially written LinkedList class.
2. Here is a list of existing functions:
   2.1 addLast - adds a new element with given value to the end of Linked List
   2.2. display - Prints the elements of linked list from front to end in a single line.
   All elements are separated by space.
2.3. size - Returns the number of elements in the linked list.
2.4. removeFirst - Removes the first element from Linked List.
3. You are required to complete the body of getFirst, getLast and getAt function
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".
4. Input and Output is managed for you.

Sample Input

```
addLast 10
getFirst
addLast 20
addLast 30
getFirst
getLast
getAt 1
```

Sample Output

```
10
10
30
20
40
20
Invalid arguments
```

you need to write code for

getfirst
getlast
getatindex

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

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

Add First In Linked List

Easy

Prev Next

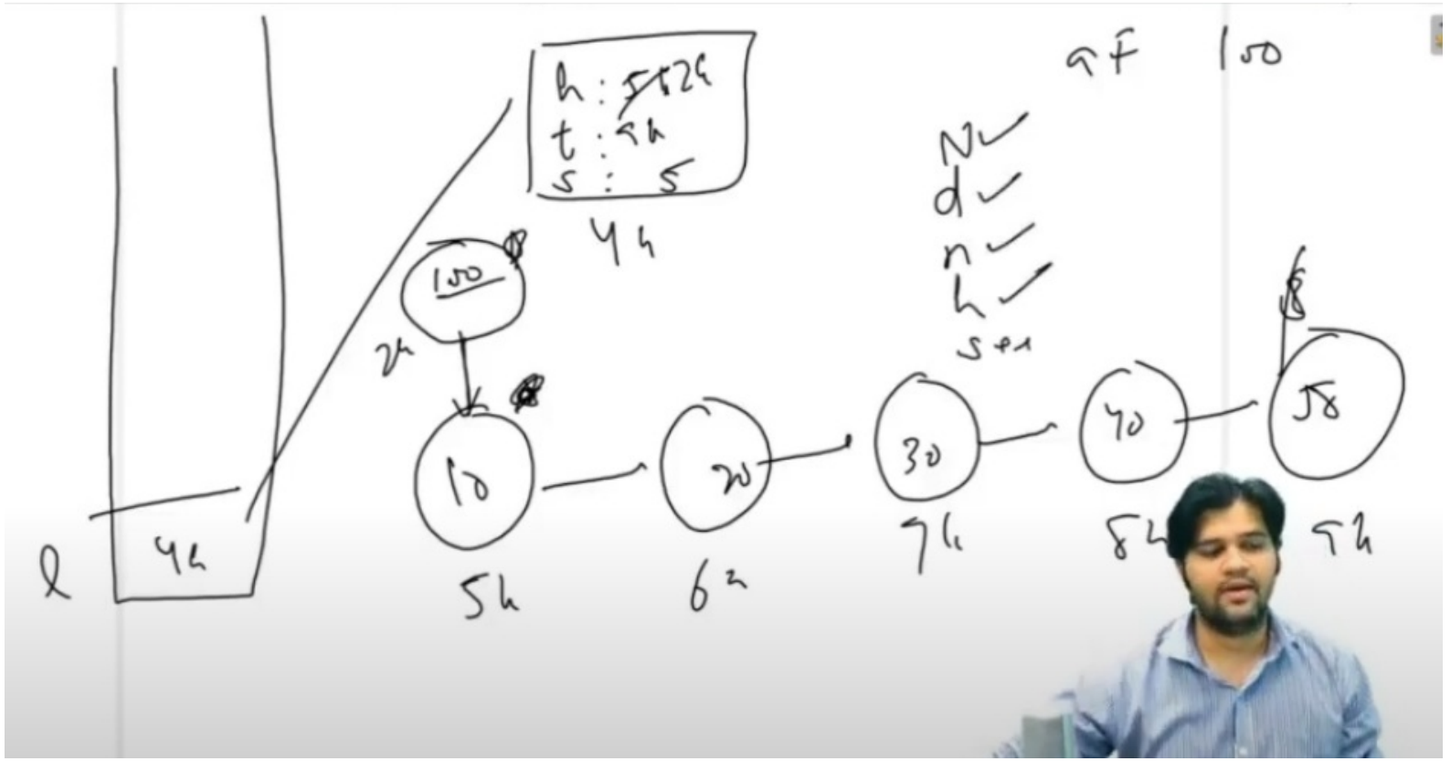
1. You are given a partially written LinkedList class.
2. Here is a list of existing functions:
   2.1 addLast - adds a new element with given value to the end of Linked List
   2.2. display - Prints the elements of linked list from front to end in a single line.
   All elements are separated by space.
   2.3. size - Returns the number of elements in the linked list.
   2.4. removeFirst - Removes the first element from Linked List.
   2.5. getFirst - Returns the data of first element.
   2.6. getLast - Returns the data of last element.
   2.7. getAt - Returns the data of element available at the index passed.
3. You are required to complete the body of addFirst function. This function should add the element to the beginning of the linkedlist and appropriately set the head, tail and size data-members.
4. Input and Output is managed for you.

Sample Input

```
addFirst 10
getFirst
addFirst 20
getFirst
getLast
display
size
```

Sample Output

```
10
20
10
20 10
2
40
20
```




```
public void addFirst(int val) {
    Node temp = new Node();
    temp.data = val;
    temp.next = head;
    head = temp;

    if(size == 0){
        tail = temp;
    }

    size++;
}
```

Add At Index In Linked List

Easy

Prev Next

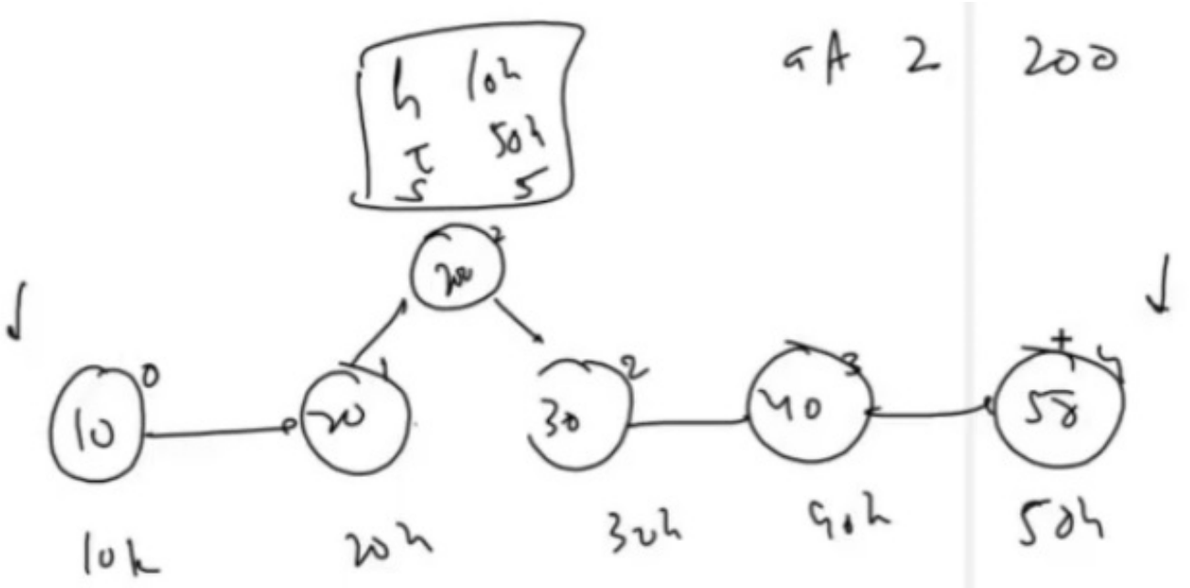
1. You are given a partially written LinkedList class.
2. Here is a list of existing functions:
   2.1 addLast - adds a new element with given value to the end of Linked List
   2.2. display - Prints the elements of linked list from front to end in a single line. All elements are separated by space
   2.3. size - Returns the number of elements in the linked list.
   2.4. removeFirst - Removes the first element from Linked List.
   2.5. getFirst - Returns the data of first element.
   2.6. getLast - Returns the data of last element.
   2.7. getAt - Returns the data of element available at the index passed.
   2.8. addFirst - adds a new element with given value in front of linked list.
3. You are required to complete the body of addAt function. This function should add the element at the index mentioned as parameter. If the idx is inappropriate print "Invalid arguments".
4. Input and Output is managed for you.

Sample Input

```
addFirst 10
getFirst
addAt 0 20
getFirst
getLast
display
size
```

Sample Output

```
10
20
10
20 10
2
40
20
```




```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 node = new Node();
      node.data = val;


      Node temp = head;
      for(int i = 0; i < idx - 1; i++){
          temp = temp.next;
      }

      node.next = temp.next;
      temp.next = node;

      size++;
  }
}
```

Remove Last In Linked List

Easy

Prev Next

1. You are given a partially written LinkedList class.
2. Here is a list of existing functions:
   2.1 addLast - adds a new element with given value to the end of Linked List
   2.2. display - Prints the elements of linked list from front to end in a single line.
   All elements are separated by space
   2.3. size - Returns the number of elements in the linked list.
   2.4. removeFirst - Removes the first element from Linked List.
   2.5. getFirst - Returns the data of first element.
   2.6. getLast - Returns the data of last element.
   2.7. getAt - Returns the data of element available at the index passed.
   2.8. addFirst - adds a new element with given value in front of linked list.
   2.9. addAt - adds a new element at a given index.
3. You are required to complete the body of removeLast function. This function should remove the las data members. If the size is 0, should print "List is empty". If the size is 1, should set both head and tail to
4. Input and Output is managed for you.

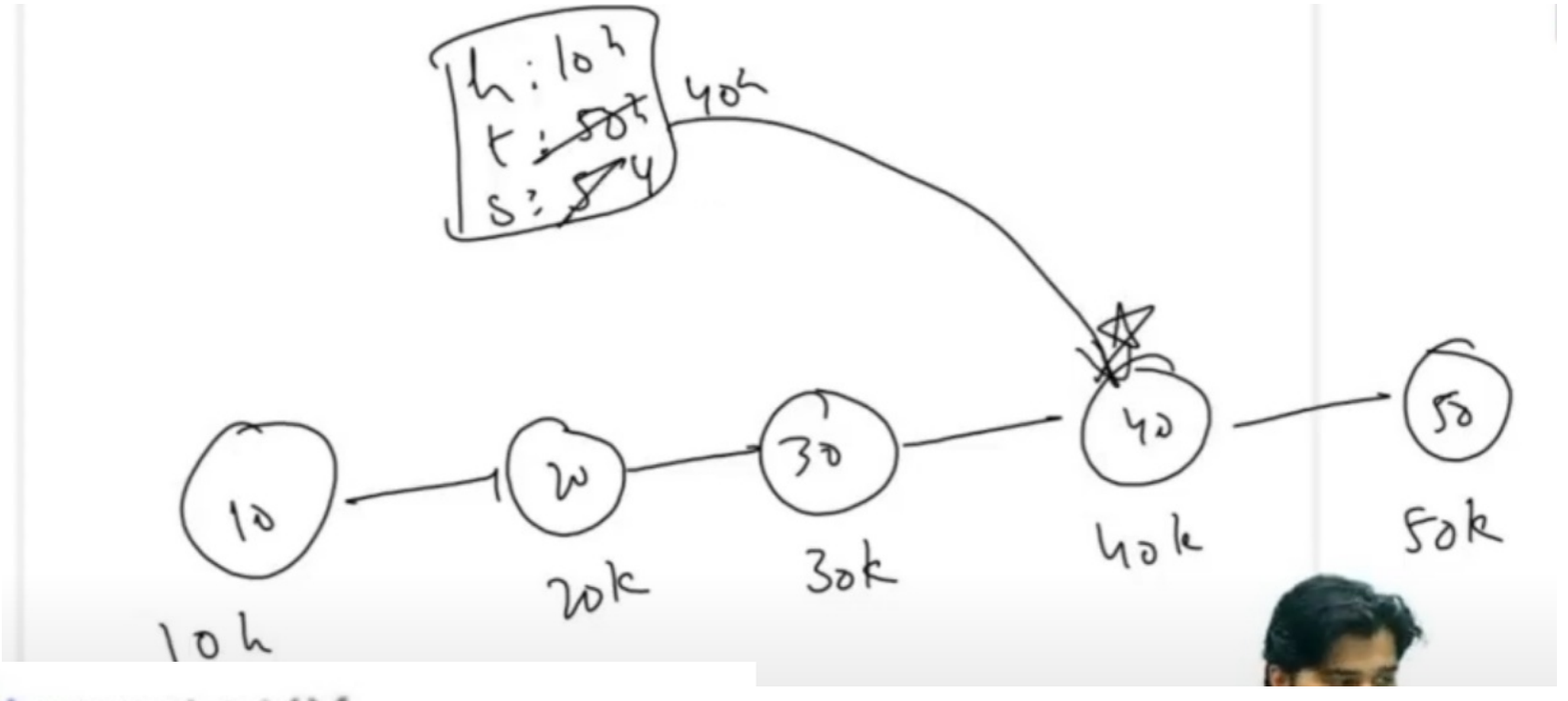
Input Format

Sample Input

```
addFirst 10
getFirst
addAt 0 20
getFirst
getLast
display
size
```

Sample Output

```
10
20
10
20 10
2
40
20
```




```java
public void removeLast(){
  if (size == 0) {
    System.out.println("List is empty");
  } else if (size == 1) {
    head = tail = null;
    size = 0;
  } else {
    Node temp = head;
    for(int i = 0; i < size - 2; i++){
        temp = temp.next;
    }

    tail = temp;
    temp.next = null;
    size--;
  }
}
```

Remove At Index In Linked List

Easy

Prev Next

1. You are given a partially written LinkedList class.
2. Here is a list of existing functions:
2.1 addLast - adds a new element with given value to the end of Linked List
2.2. display - Prints the elements of linked list from front to end in a single line. All elements are separated by space
2.3. size - Returns the number of elements in the linked list.
2.4. removeFirst - Removes the first element from Linked List.
2.5. getFirst - Returns the data of first element.
2.6. getLast - Returns the data of last element.
2.7. getAt - Returns the data of element available at the index passed.
2.8. addFirst - adds a new element with given value in front of linked list.
2.9. addAt - adds a new element at a given index.
2.10. removeLast - removes the last element of linked list.
3. You are required to complete the body of removeAt function. The function should remove th as parameter. If the size is 0, should print "List is empty". If the index is inappropriate print "Invali hen list has a single element.
4. Input and Output is managed for you.

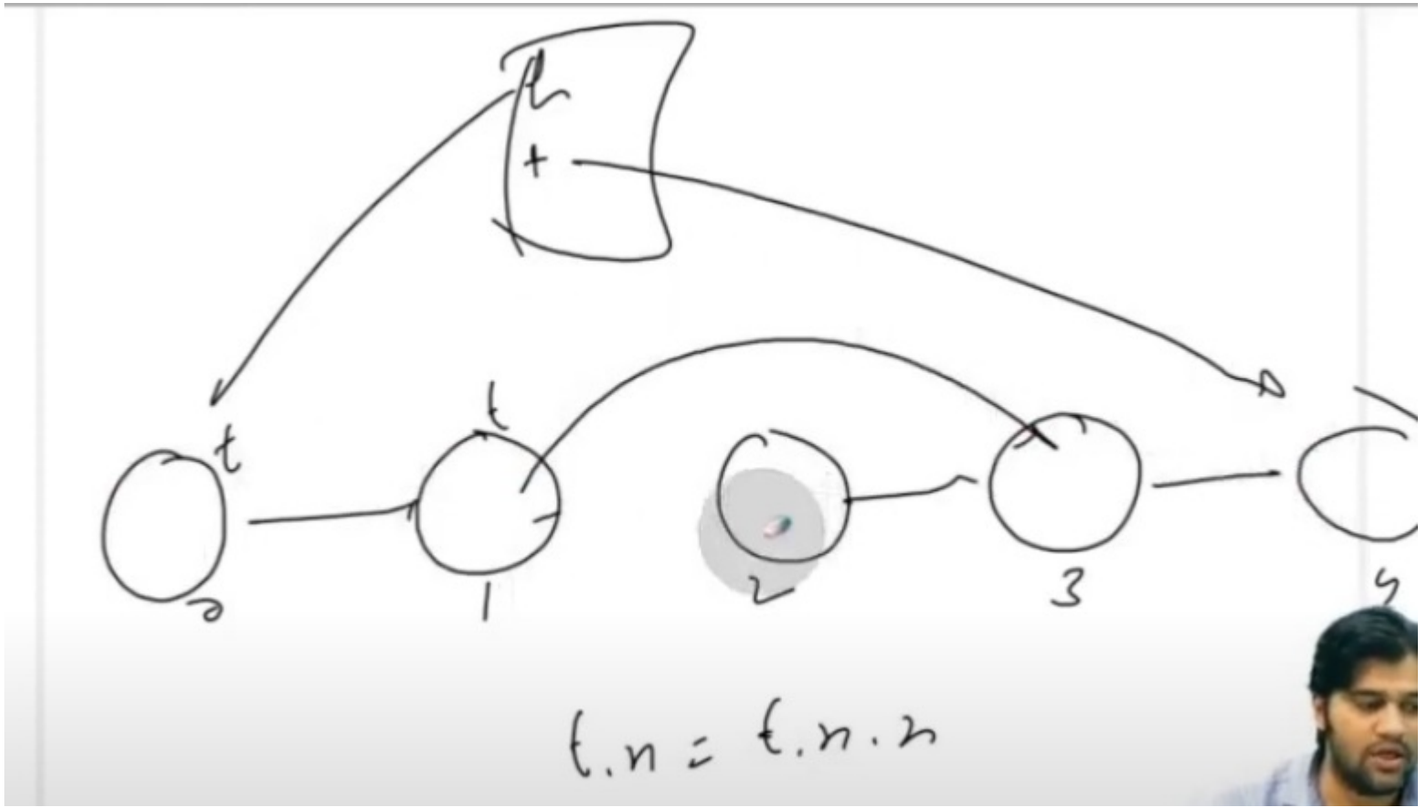
Input Format

Sample Input

```
addFirst 10
getFirst
addAt 0 20
getFirst
getLast
display
size
```

Sample Output

```
10
20
10
20 10
2
40
30
```




```
public void removeAt(int idx) {
  if(idx < 0 || idx >= size){
      System.out.println("Invalid arguments");
  } else if(idx == 0){
      removeFirst();
  } else if(idx == size - 1){
      removeLast();
  } else {
      Node temp = head;
      for(int i = 0; i < idx - 1; i++){
          temp = temp.next;
      }

      temp.next = temp.next.next;
      size--;
  }
}
```