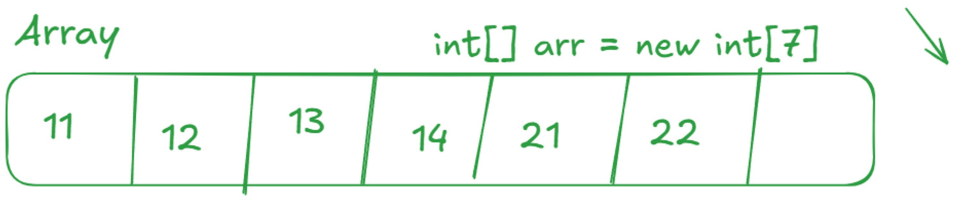
Introduction To LinkedList & Traversal [Day 13]

07 October 2024 15:52

Introduction To LinkedList & Traversal [Day 13]

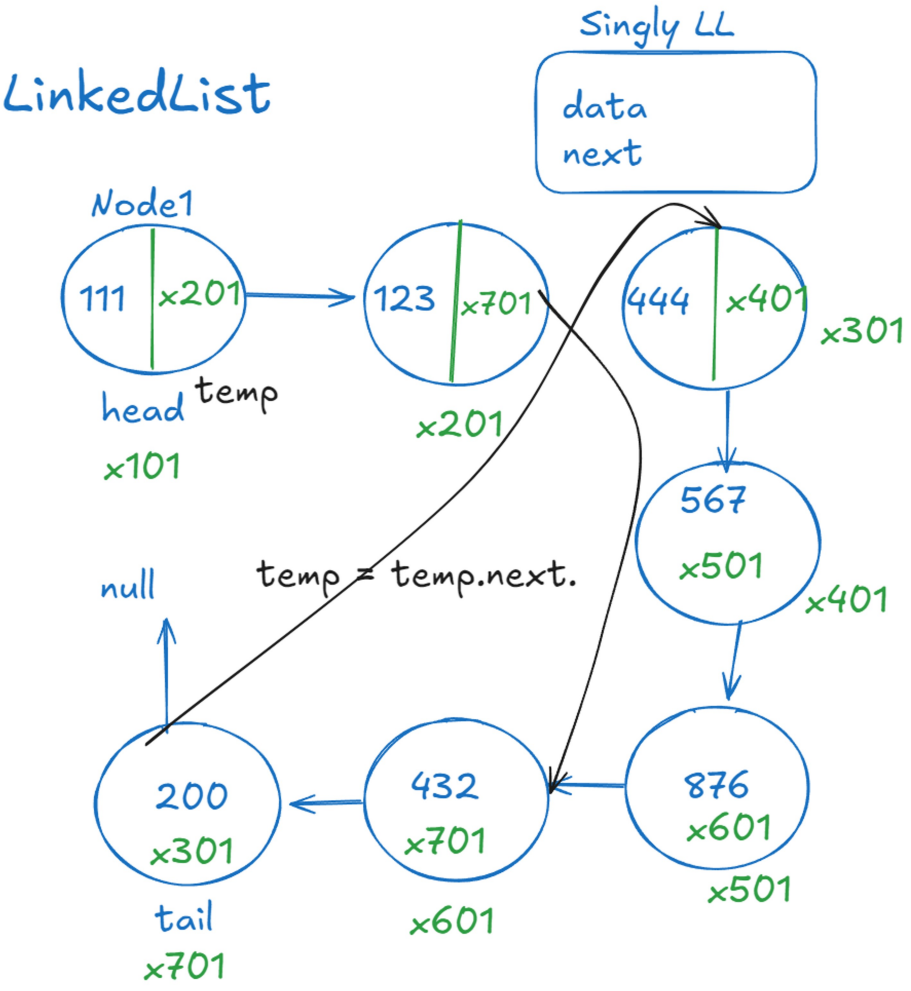


Limitation :

1. Fixed Size.

2. If I wanted to add or remove element in between
   Time Complexity : O(N)

3. Memory Waste : int[] arr = new int[1000].
   --> 200



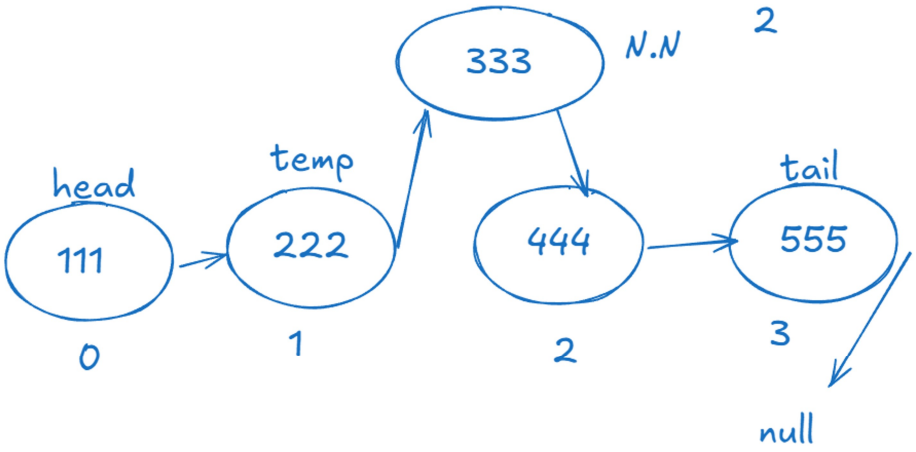
x701

temp

111 --> 123 --> 432 --> 200 --> null

```
while (temp != null){
    temp = temp.next.
}
```

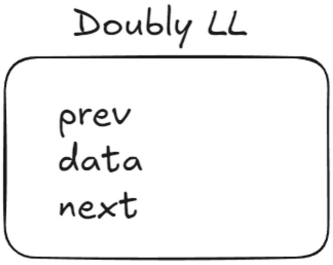
```
for(int i = 0 ; i < arr.length ; i++){

}
```

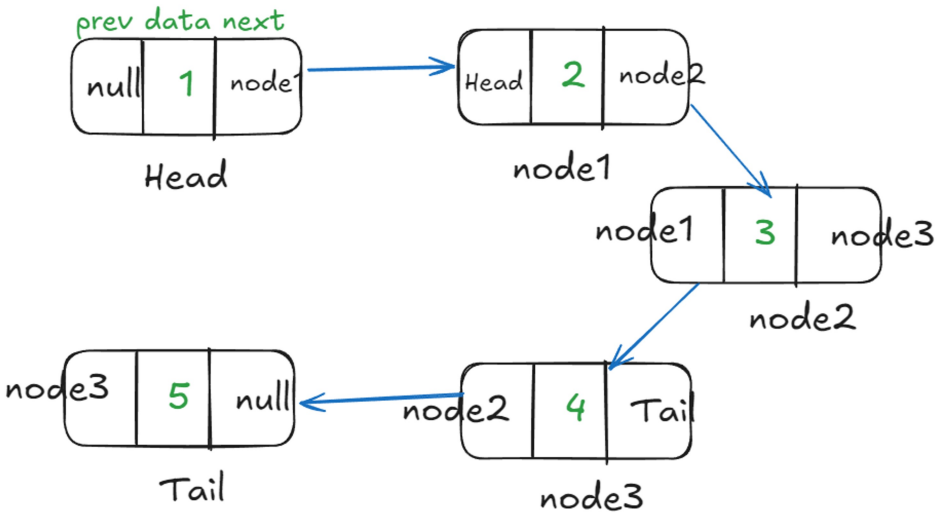


Types Of LinkedList

1. Singly LinkedList

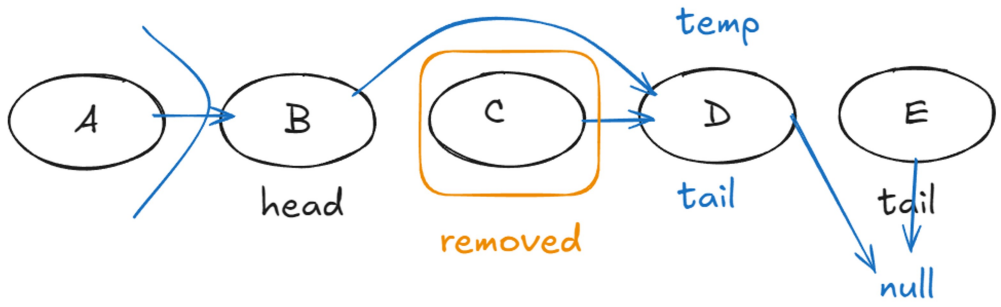
2. Doubly LinkedList





3. Circular LinkedList

Remove An Element in LL.



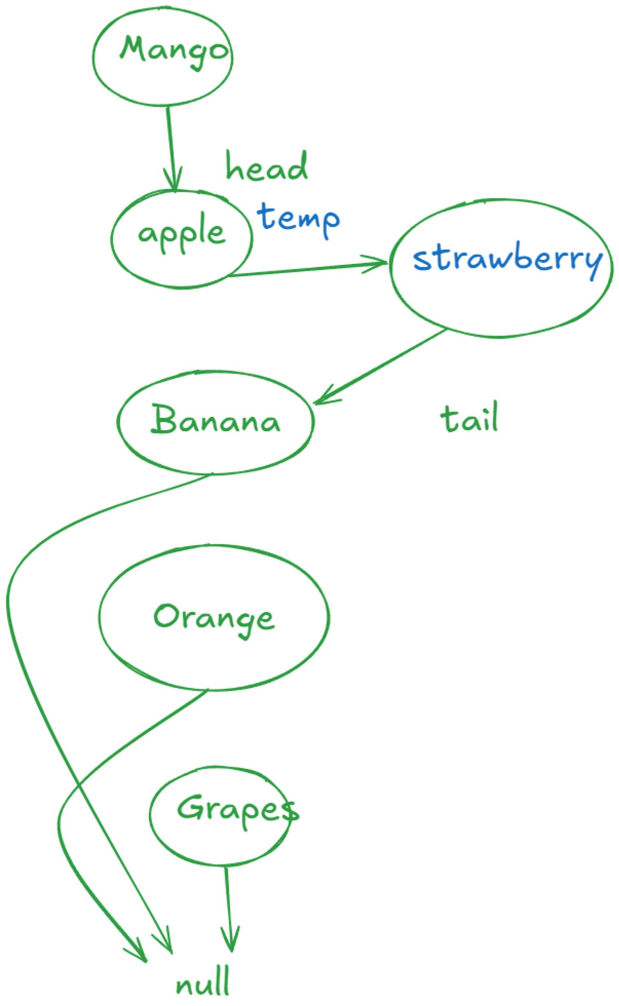
temp = temp.next

temp.next.next == null

```
import java.util.*;
public class LinkedListExample {
```

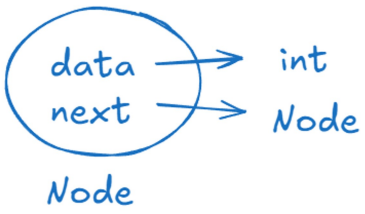
Mango

```java
import java.util.*;
public class LinkedListExample {
    public static void main(String[] args) {
        LinkedList<String> list = new LinkedList<>();

        //1. Adding Element in a LinkedList
        list.add("Apple");
        list.add("Banana");
        list.add("Orange");
        System.out.print("LinkedList : " + list);
        System.out.println();

        // 2. Adding elemenets at specific position.
        list.addFirst("Mango");
        System.out.print("LinkedList : " + list);
        System.out.println();
        list.addLast("Grapes");
        System.out.print("LinkedList : " + list);
        System.out.println();
        list.add(2 , "Pineapple");
        System.out.print("LinkedList : " + list);
        System.out.println();

        //3. Accessing elements :
        System.out.println(list.getFirst());
        System.out.println(list.getLast());
        System.out.println(list.get(3));

        //4. Removing Element
        list.removeFirst();
        System.out.print("LinkedList : " + list);
        System.out.println();
        list.removeLast();
        System.out.print("LinkedList : " + list);
        System.out.println();
        list.remove(3);
        System.out.print("LinkedList : " + list);
        System.out.println();

        //5. Check if LL contain element or not
        System.out.println(list.contains("Banana"));
        System.out.println(list.contains("Mango"));

        //6. Updating the element :
        System.out.println(list.set(1,"Strawberry"));
        System.out.print("LinkedList : " + list);
        System.out.println();

        //7. Iterating through the LL
        for(String fruit : list){
            System.out.println(fruit);
        }
```

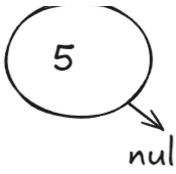


```
        for(String fruit : list){
            System.out.println(fruit);
        }

        //8. Using Iterator
        Iterator<String> iterator = list.iterator();
        while(iterator.hasNext()){
            System.out.println(iterator.next());
        }
        System.out.println(list.size());

        // 9. Clearing the linkedlist
        list.clear();
        System.out.print("LinkedList : " + list);
        System.out.println();

        //10. isEmpty :
        System.err.println(list.isEmpty());

        // 11. Size of LL:
        System.out.println(list.size());
    }
}
```

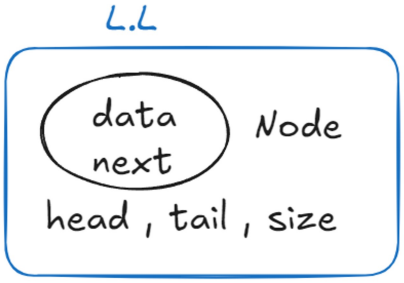
Implementation of a LinkedList



Create a node of data = 5.




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

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

    private Node head;
    private Node tail;
    private int size;

}
```



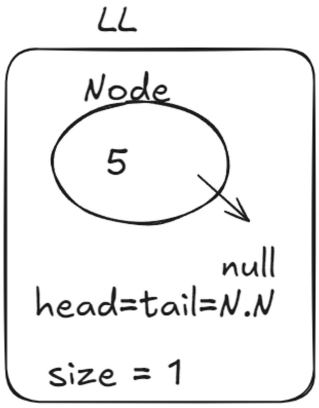
Inserting At End

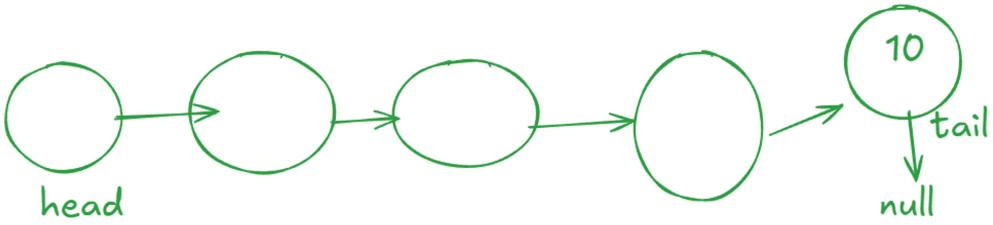
LL is Empty

head

null

tail





tail.next = new Node,
tail = newNode

size++

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

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

    private Node head;
    private Node tail;
    private int size;
    public SinglyLinkedList(){
        head = null;
        tail = null;
        size = 0;
    }

    public void insertAtEnd(int data){
        Node newNode = new Node(data);
        if(head == null){
```

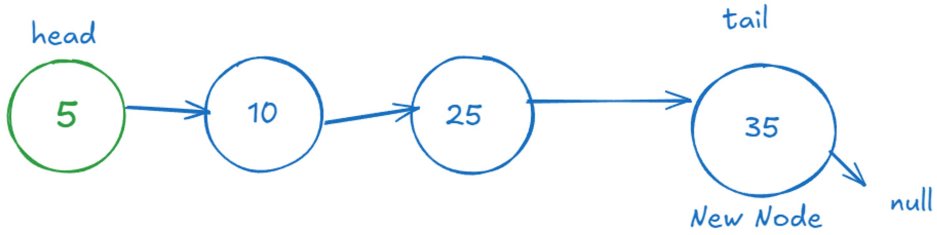
```java
public void insertAtEnd(int data){
    Node newNode = new Node(data);
    if(head == null){
        head = newNode;
        tail = newNode;
    }else{
        tail.next = newNode;
        tail = newNode;
    }
    size++;
}
```

```java
public static void main(String[] args) {
    SinglyLinkedList list = new SinglyLinkedList();
    list.insertAtEnd(data:5);
    list.insertAtEnd(data:10);
}
```

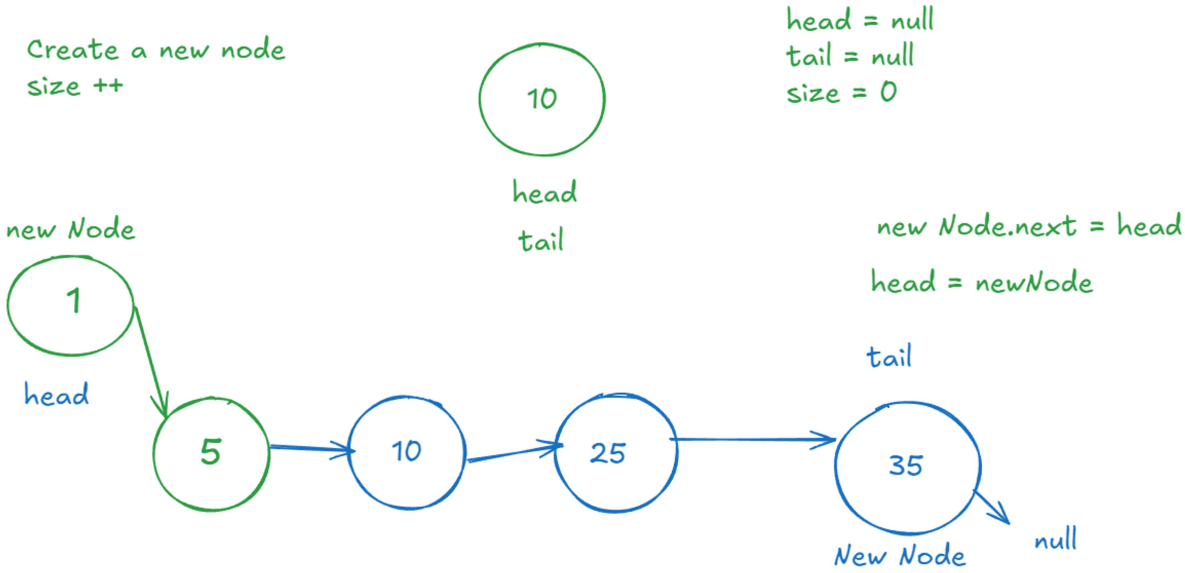
```java
SinglyLinkedList list = new SinglyLinkedList();
list.insertAtEnd(data:5);
list.insertAtEnd(data:10);
System.out.println(head);
System.out.println(head.data);
System.out.println(tail);
System.out.println(head.next);
System.out.println(tail.data);
```

```
List'
SinglyLinkedList$Node@5acf9800
5
SinglyLinkedList$Node@4617c264
SinglyLinkedList$Node@4617c264
10
```

because head.next == tail



Insert At Beginning :

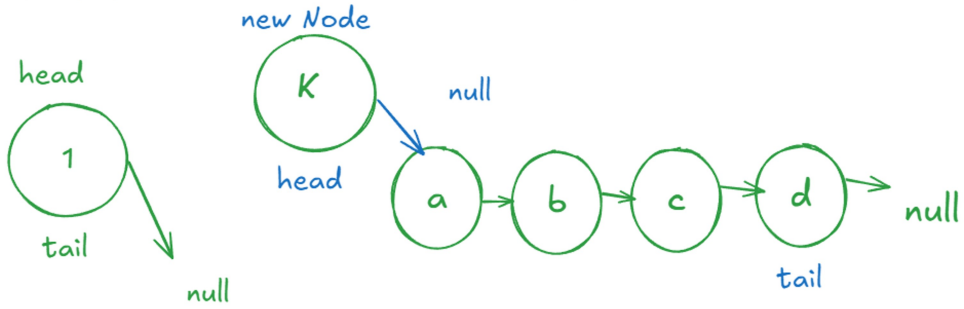


head

1 --> 5 --> 10 --> 25 --> 35 --> null

tail

```java
public void inserAtBeginning(int data){
    Node newNode = new Node(data);
    if(head == null){
        head = newNode;
        tail = newNode;
    }else{
        newNode.next = head;
        head = newNode;
    }
    size++;
}
```



K --> a --> b --> c --> d --> null

Insert a new Node At random index in LinkedList

1. Insert at idx -- 0 : InsertAtBeginning(data)

2. Insert at idx -- size : InsertAtEnd(data)

Insert a new Node At random index in LinkedList

1. Insert at idx -- 0 : InsertAtBeginning(data)

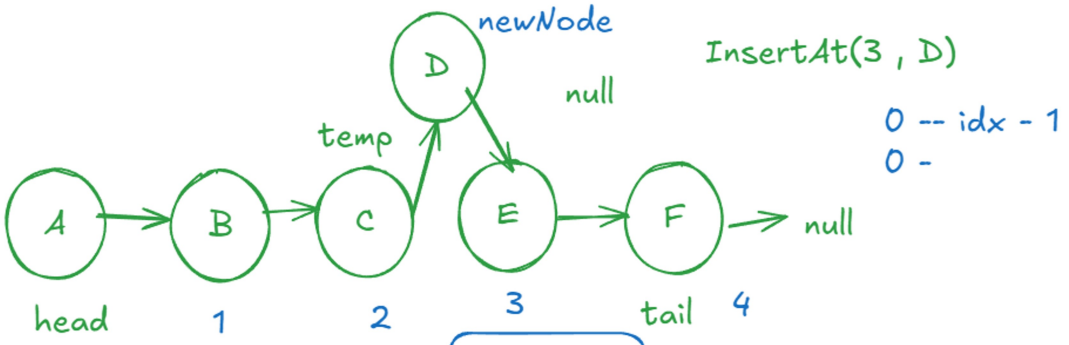
2. Insert at idx -- size : InsertAtEnd(data)

head
tail

3. idx < 0 || idx > size -> throw error.

4. Random Index between head & tail

-- idx - 1 ::: connecting the addresses.



InsertAt(3 , D)

0 -- idx - 1

0 -

temp.next

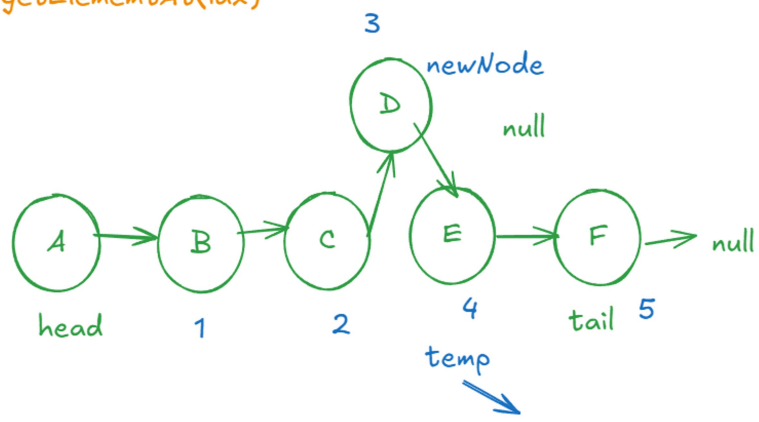
2

```
for(int i = 0; i<index - 1; i++){
    temp = temp.next;
}
```

3-1 = 2

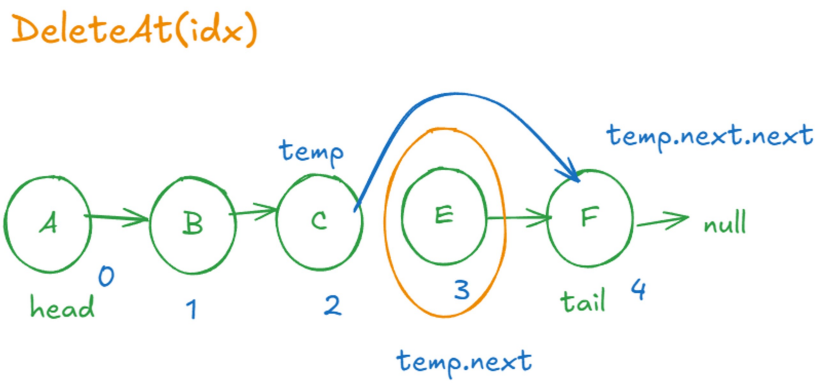
0 , 1

getElememtAt(idx)




```java
public int getElementAt(int index){
    if(index < 0 || index > size){
        throw new IndexOutOfBoundsException(s:"Index Invalid");
    }
    Node temp = head;
    for(int i = 0 ; i < index; i++){
        temp = temp.next;
    }
    return temp.data;
}
```

```java
public void display(){
    Node temp = head;
    while(temp!= null){
        System.out.print(temp.data + " ");
        temp = temp.next;
    }
    System.out.println(x:"null");
}
```



Change Status

Completed

Attendance Code: 33E4007E