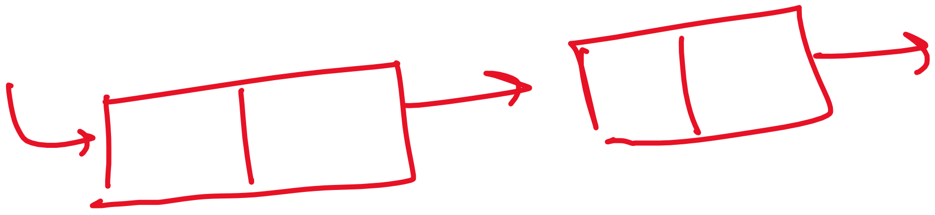
Singly Linked List

06 February 2024 10:20

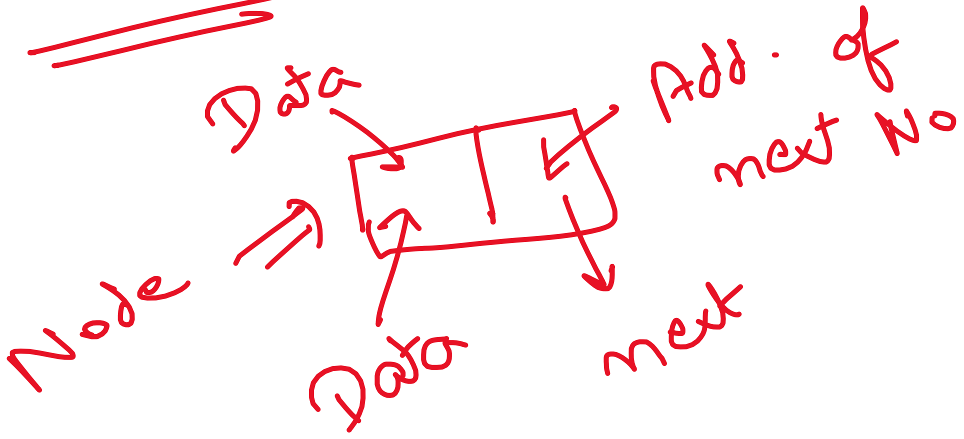
Linked List



Types

- Singly L.L
- Doubly L.L
- Circular L.L

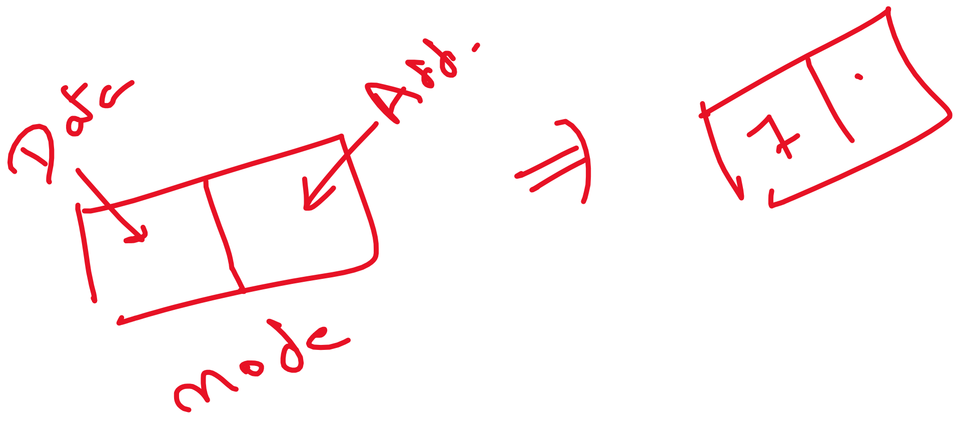
Singly L.L

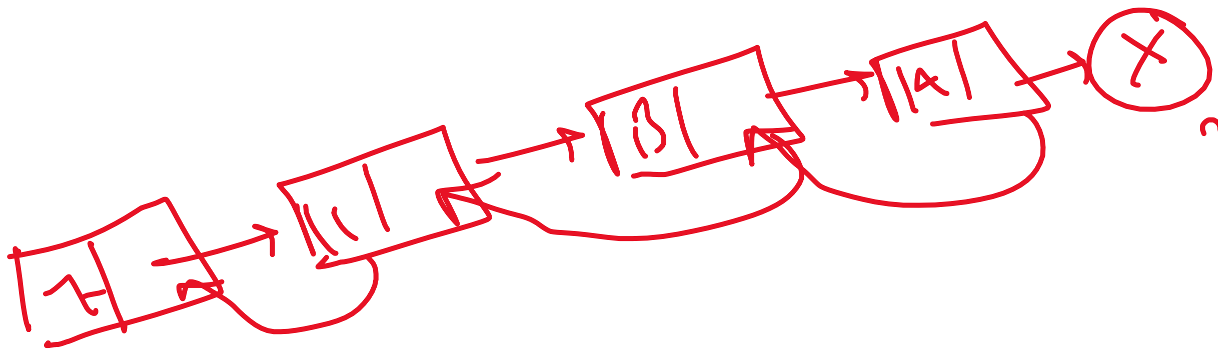


```
class node {
    int data;
    [illegible]
```

node next;

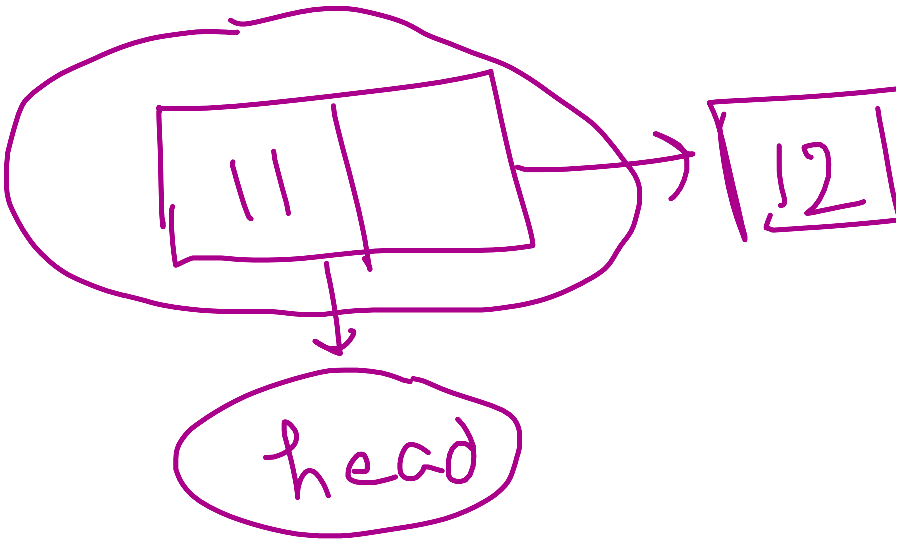
}





class node {
  int data;
  node next;

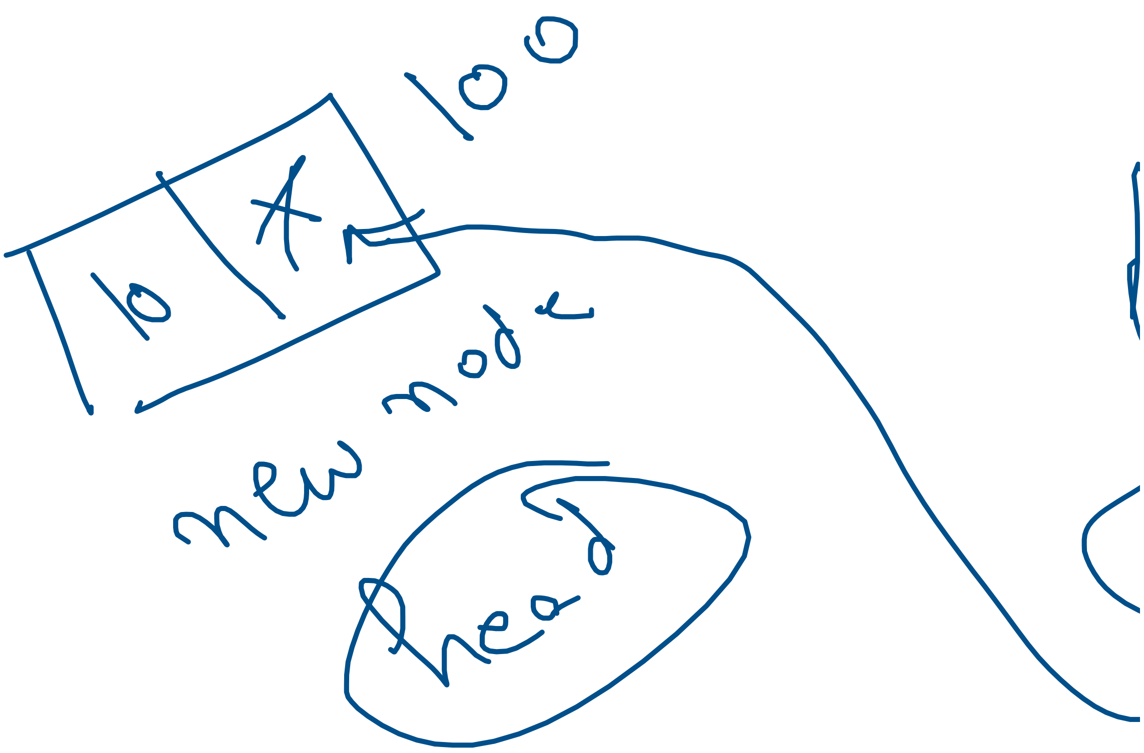
```
// constru          valu
node(v){
      this. data =
      this. ne
}
```



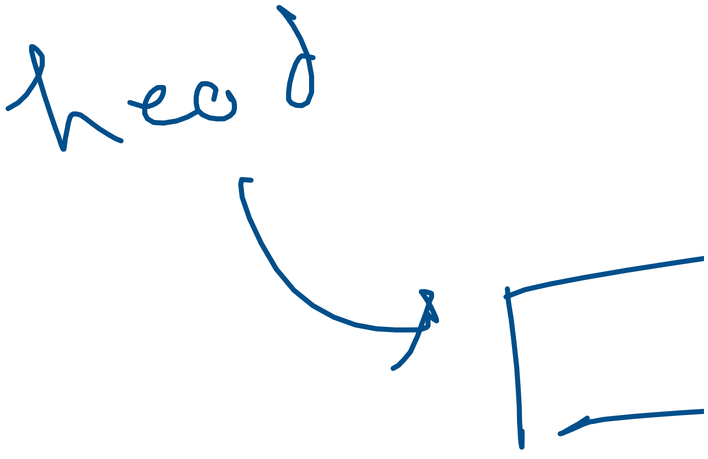
1 new node c

2 new node

3 head =

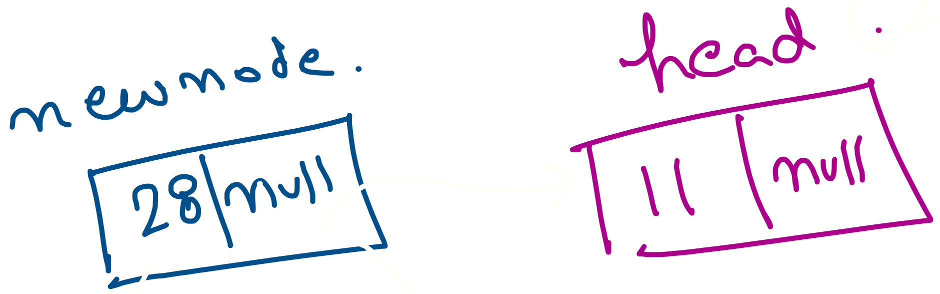


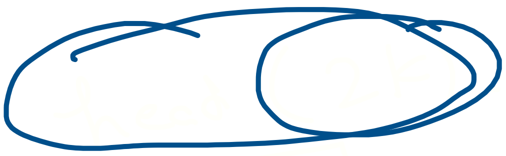
10
X
100
new node
head

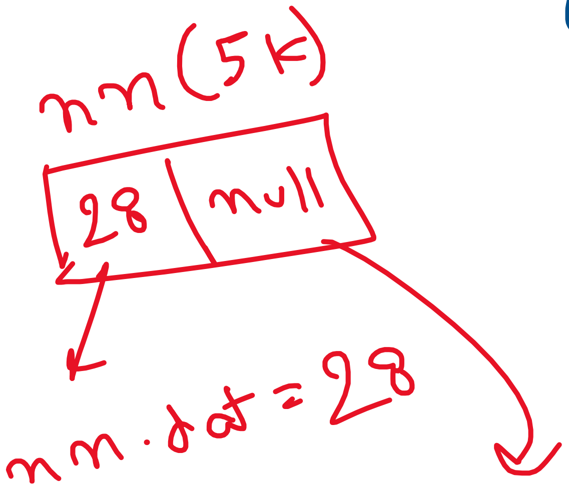


head

1) new



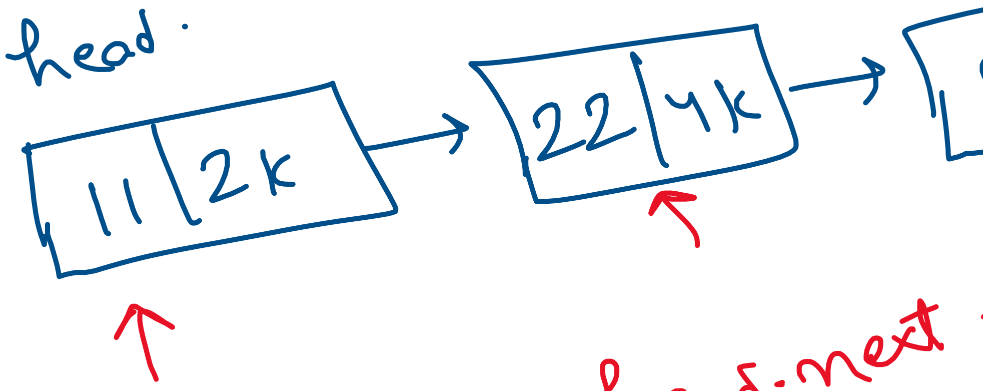




xt = ~~null~~ 2K;

nn.ne

head = ~~5k~~ (nn)



head.next

temp.dat

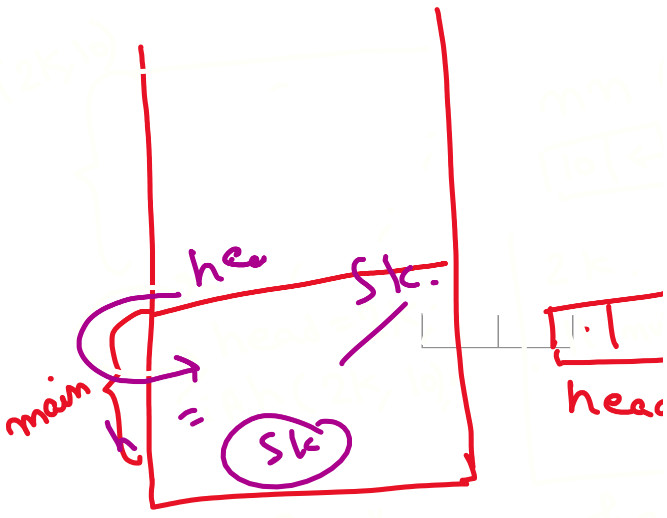
traverse (he

node temp

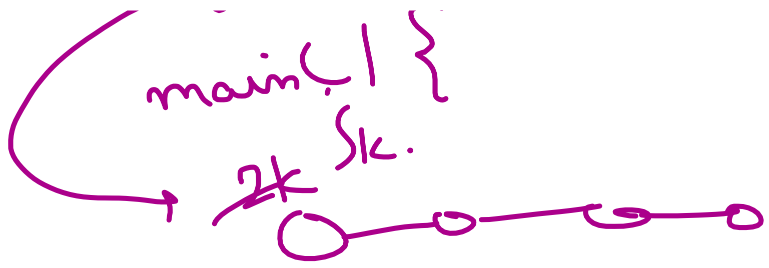
hile (t

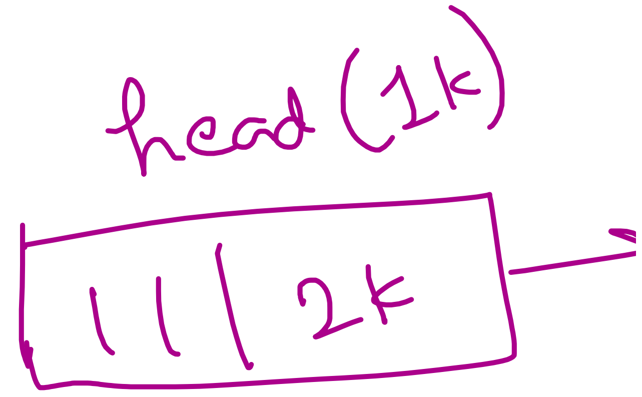
wha

```java
import java.util.*;

public class Main {
  static class node{
    int data;
    node next;

    //constructor
    node(int val){
      this.data = val;
      this.next = null;
    }
  }

  public static node insertAthead(node head,int val){
    //task1 : new node;

    node newnode = new node(val);

    //task2 : new node ke next mein head daal do;

    newnode.next = head;

    //task 3 : update head;

    head = newnode;

    return head;
  }

  public static void traverse(node head){
    node temp = head;

    while(temp != null){
      System.out.print(temp.data + "->");
      temp = temp.next;
    }
  }
  public static void main(String[] args) {
    node head = new node(11);

    insertAthead(head,10);
    insertAthead(head,12);
    insertAthead(head,12);
    insertAthead(head,15);

    traverse(head);


  }
}
```



sk



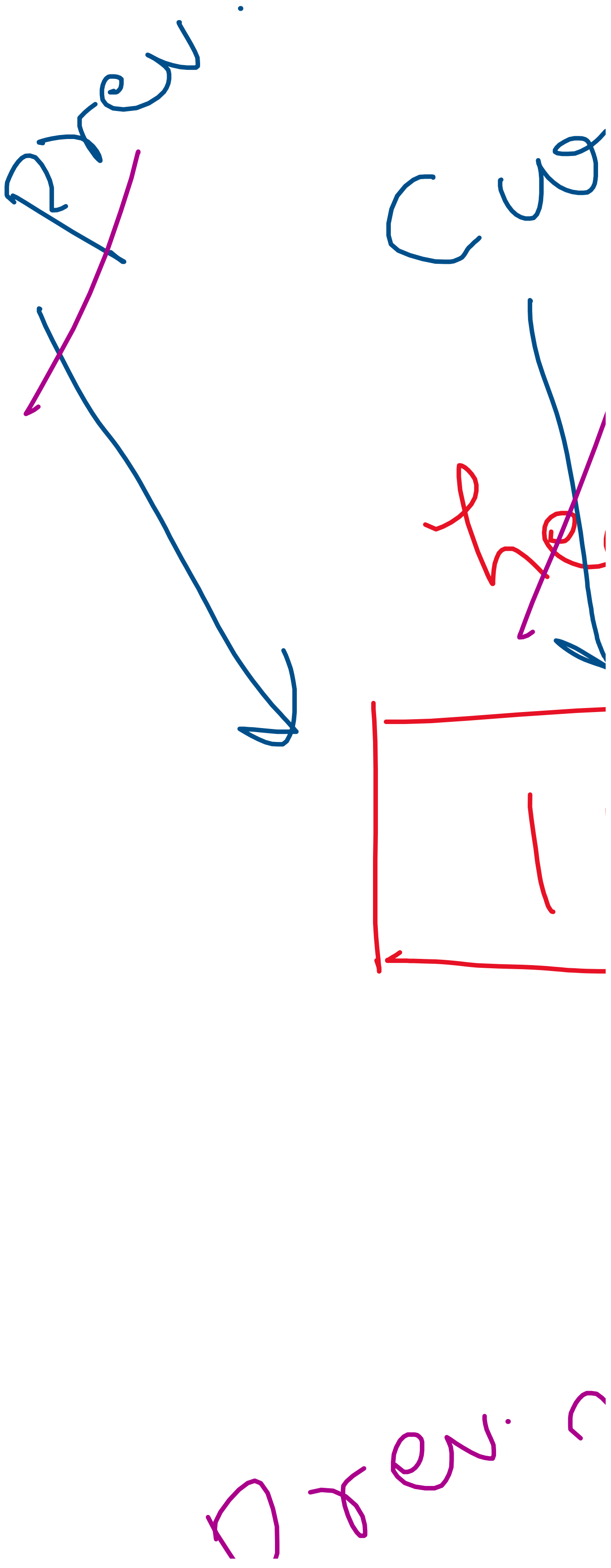


1

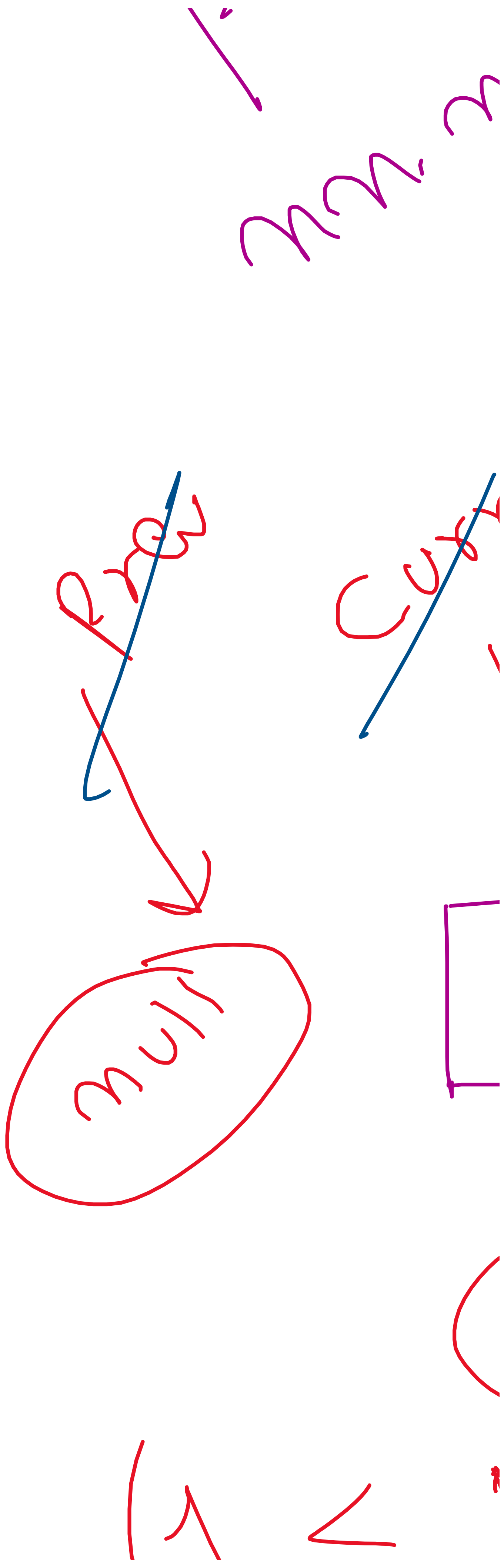
2

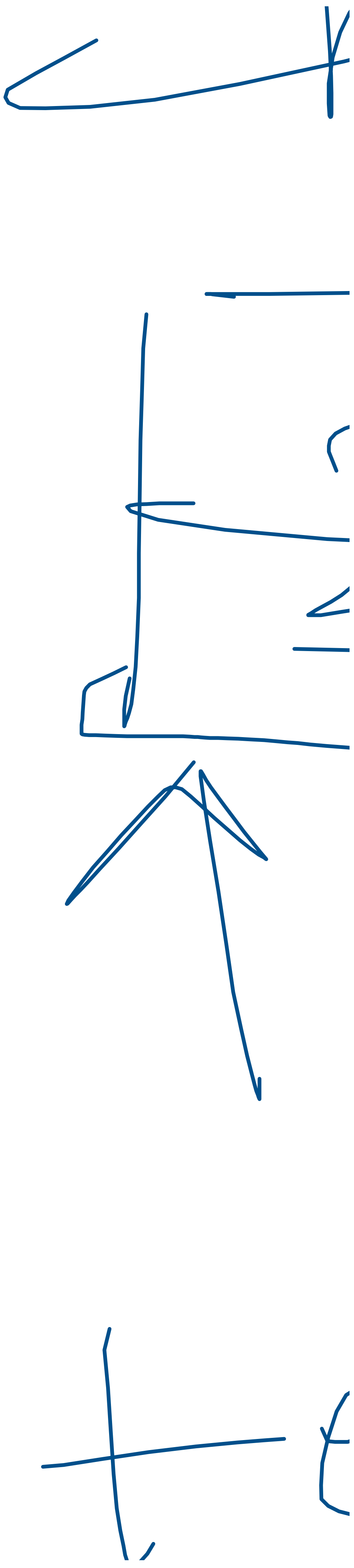
hea

$\boxed{1}$



Prev.
Cu
he
prev. n


Prev
Cur
null



#

#

null
5
null

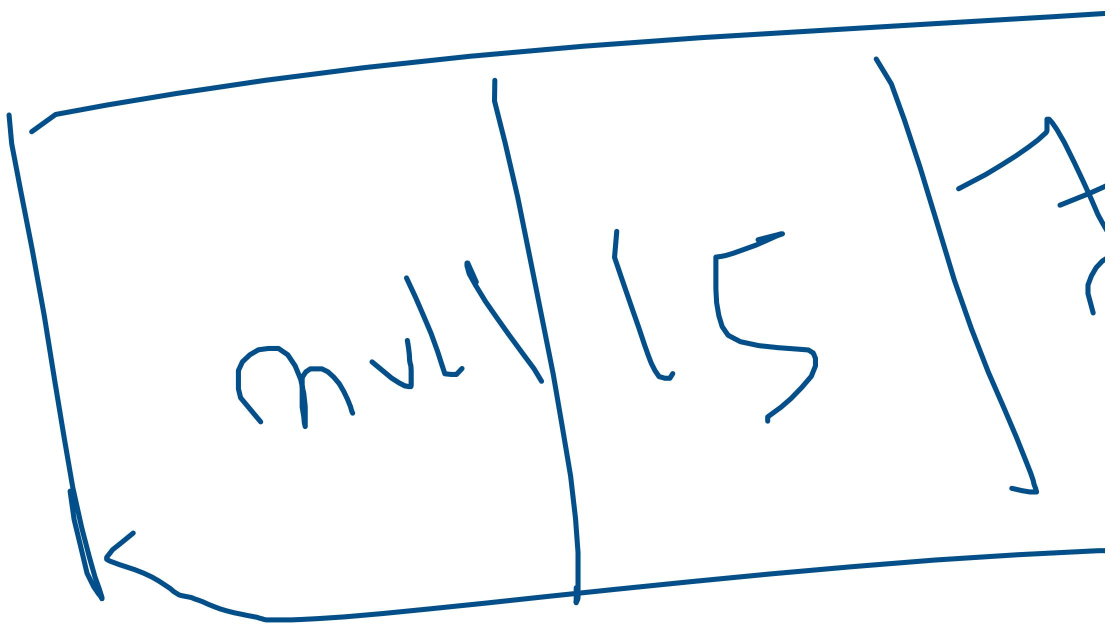
nm

(16K

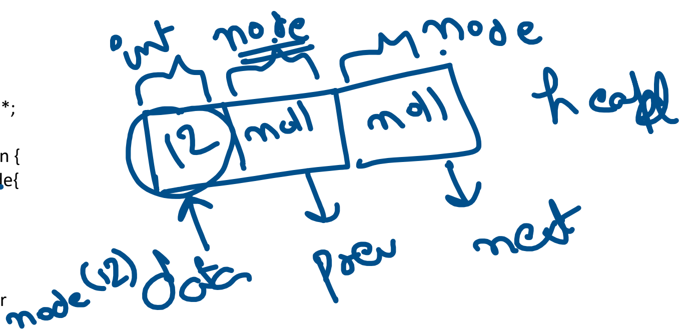
(

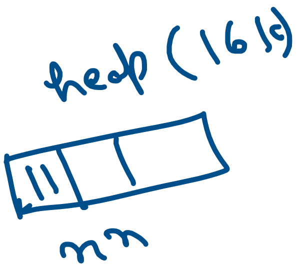
hec

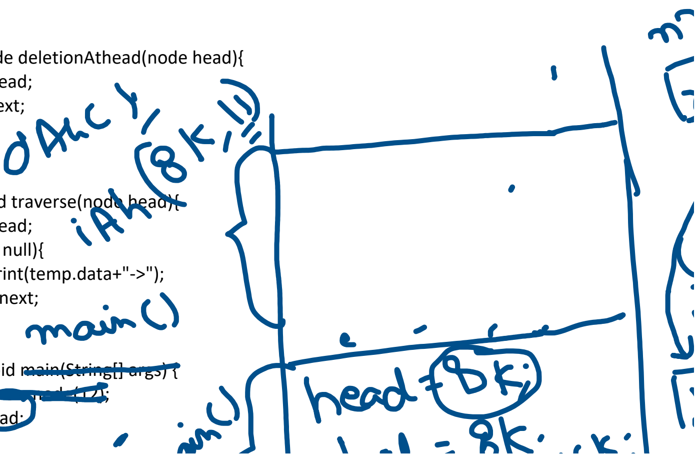
head
newnode



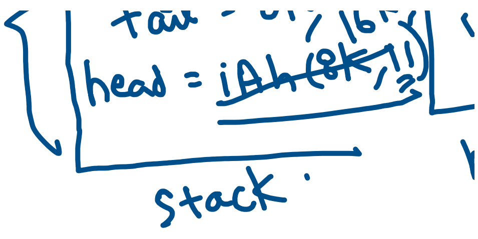
```
import java.util.*;

public class Main {
 static class node{
 int data;
  node prev;
  node next;

  ///constructor
  node(int val){
   this.data = val;
   this.next = null;
   this.prev = null;
  }
 }

 public static node insertATHead(node head,int val){
  if(head == null){
   head = new node(val);
   return head;
  }
  node newnode = new node(val);
  newnode.next = head;
  head.prev = newnode;
  head = newnode;
  return head;

 }

 public static node insertATtail(node tail,int val){
  node newnode = new node(val);
  tail.next = newnode;
  newnode.prev = tail;
  tail = newnode;
  return tail;
 }

 public static node deletionAthead(node head){
  node temp = head;
  head = head.next;
  temp = null;
  return head;
 }
 public static void traverse(node head){
  node temp = head;
  while (temp != null){
   System.out.print(temp.data+"->");
   temp = temp.next;
  }
 }
  public static void ~~main(String[] args)~~ {
   node head = new node(12);
   node tail = head;
```



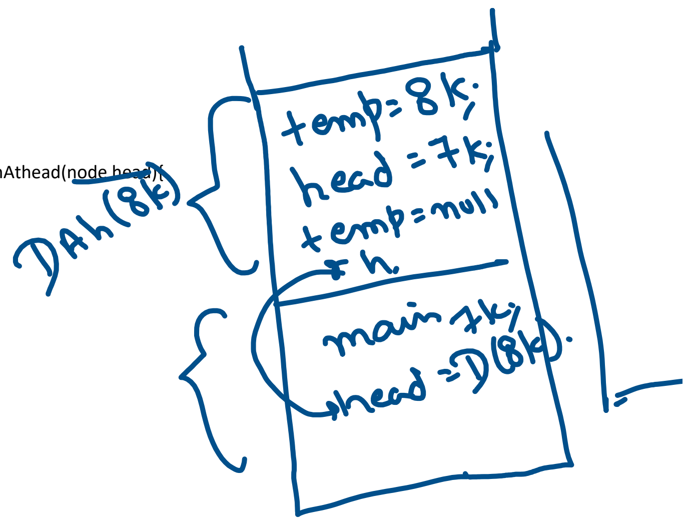


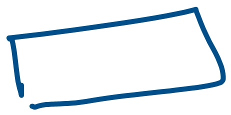


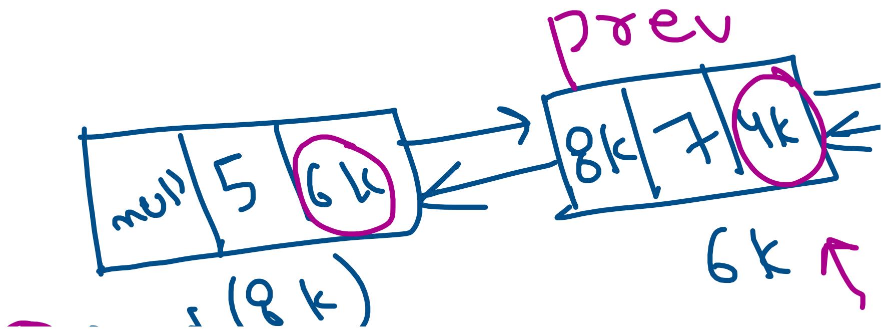
```
        head = insertATHead(head,11);
        head = insertATHead(head,10);
        head = insertATHead(head,9);
        head = insertATHead(head,8);

        traverse(head);
        System.out.println();

        tail = insertATtail(tail,13);
        tail = insertATtail(tail,14);
        tail = insertATtail(tail,15);
        tail = insertATtail(tail,16);
        traverse(head);
        System.out.println();

        head = deletionAthead(head);
        traverse(head);
        System.out.println();

         head = deletionAthead(head);
        traverse(head);
        System.out.println();

      }
    }
```

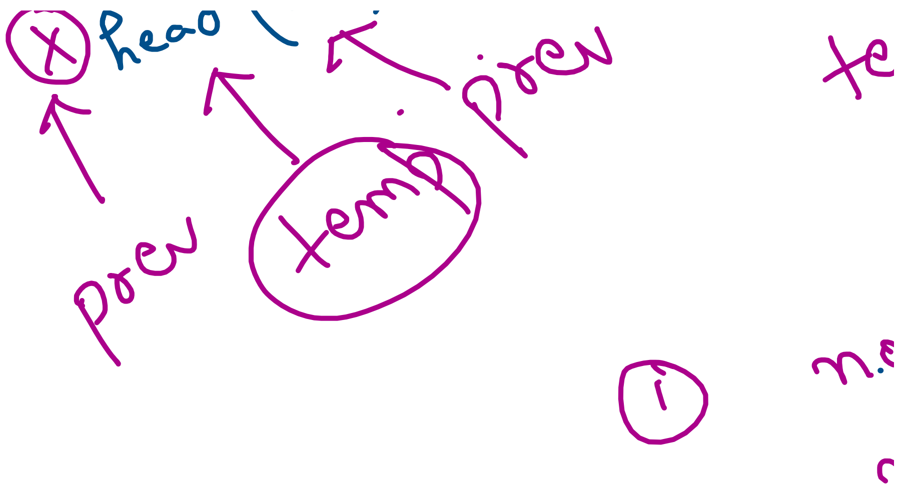


```
public static node deletionAthead(node head){
   node temp = head;
   head = head.next;
   temp = null;
   return head;
 }
```



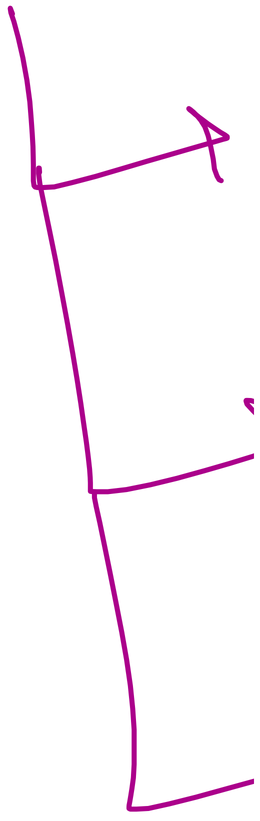




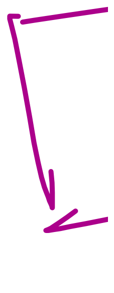


Circular

(Slow & ~~fast~~)



1



$\boxed{2}$

$\boxed{3}$