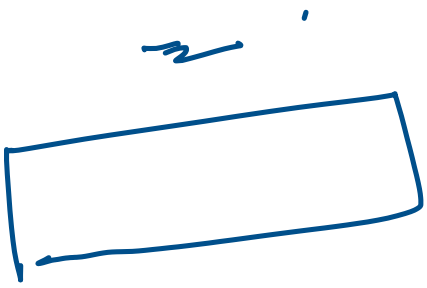
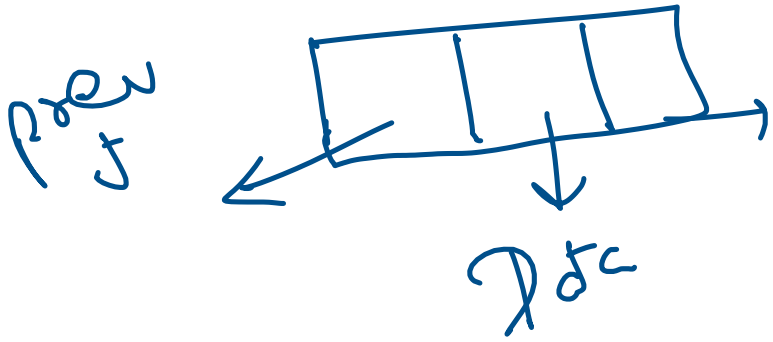
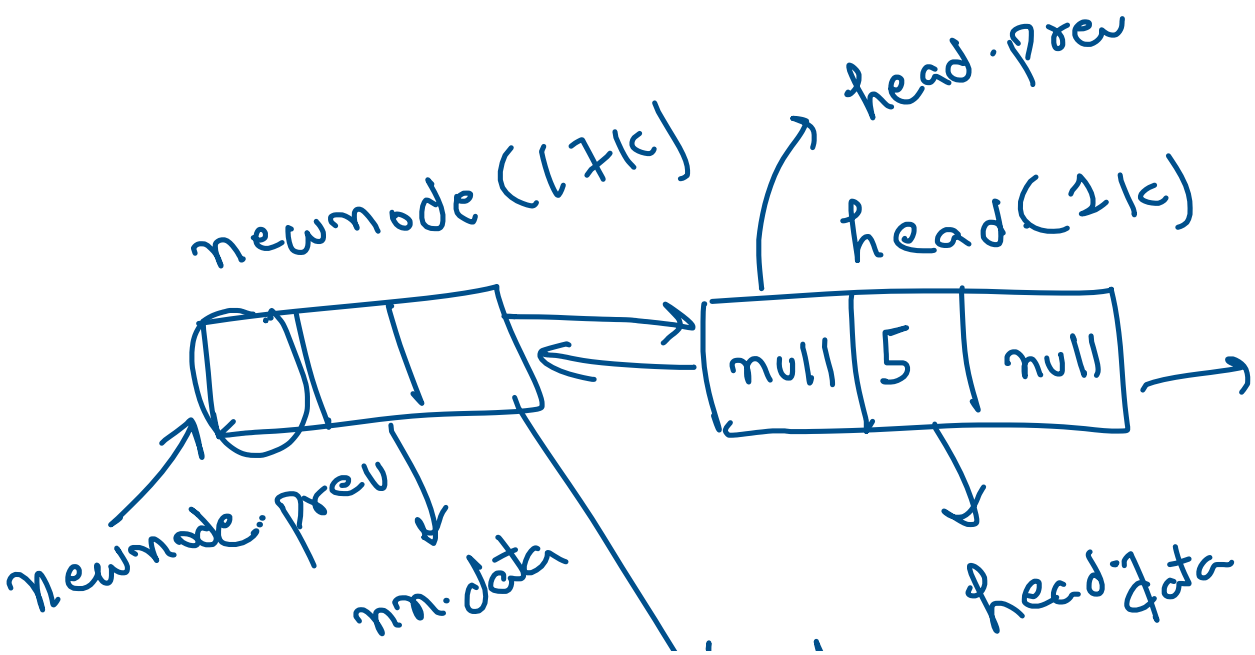
15 February 2024 14:08

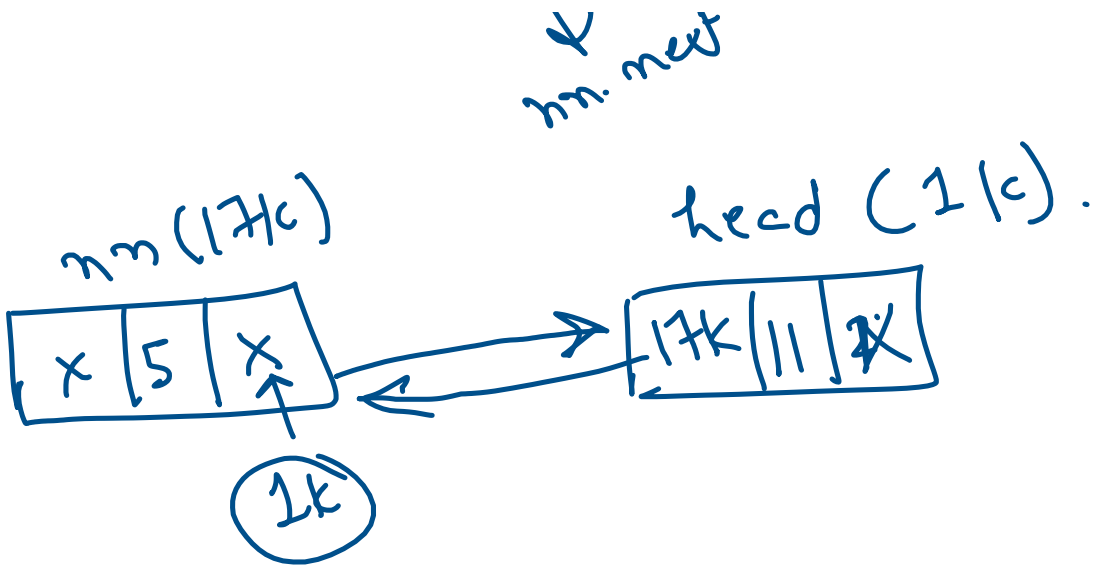


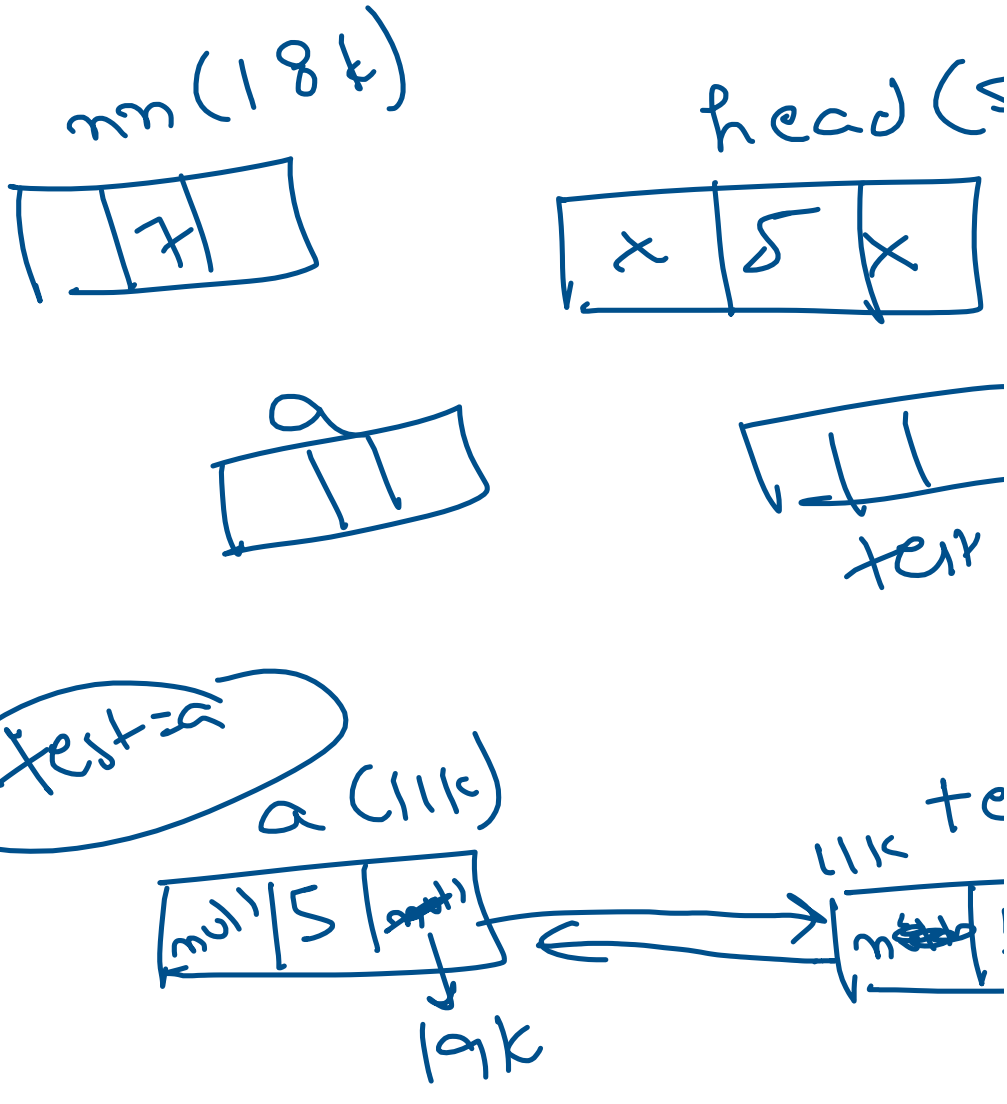
Doubly L.L

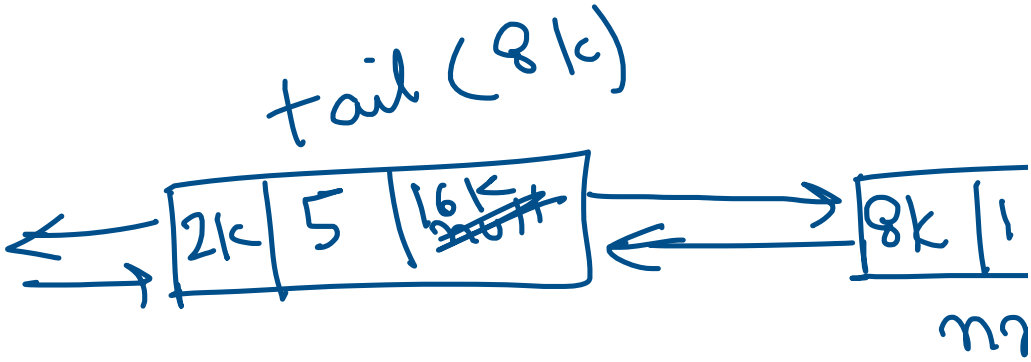
- Class Node
- Insertion
- Deletion
- Traversal



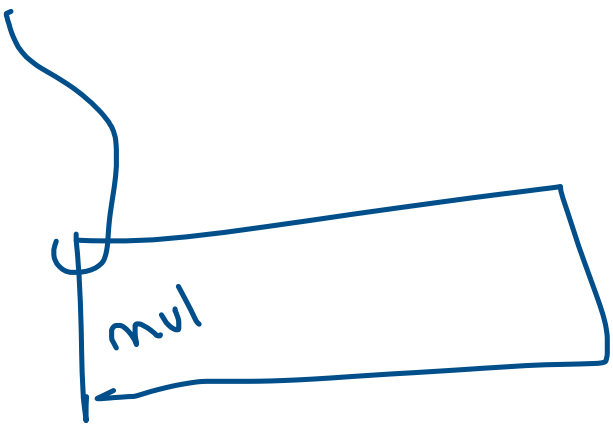



nn.next
nn (17c)
head (11c).
x
5
x
17k
11
2k


nn (18k)
7
head (
x
5
x
a
test
test=a
a (111c)
null
5
19k
111k

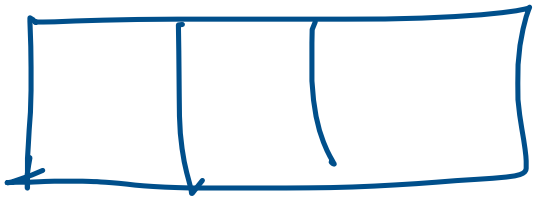


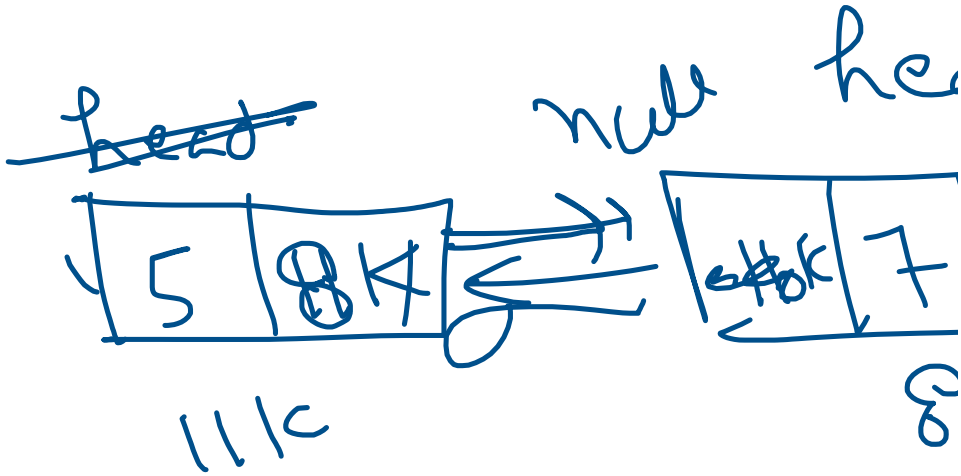
1. new node
2. tail.next = nn
3. newnode.pre
4. tail = new
5. return ta

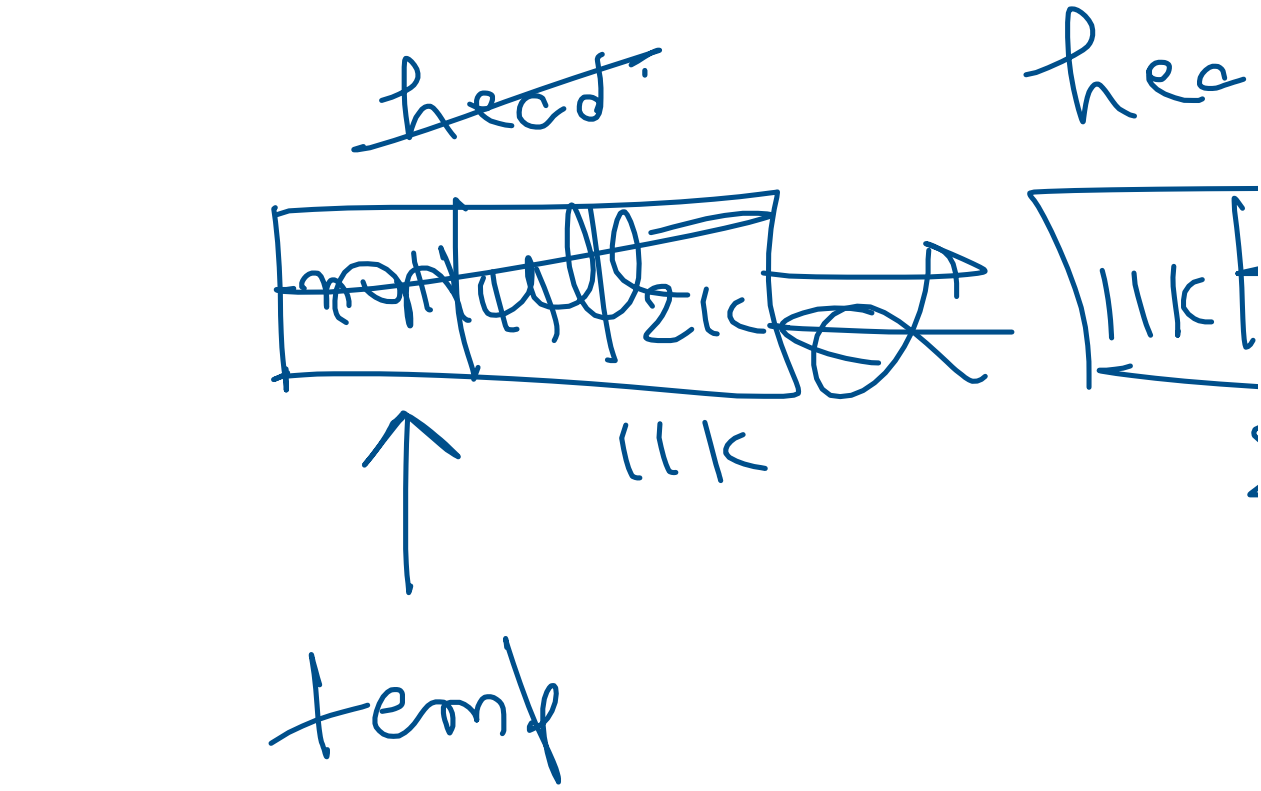


① head = h

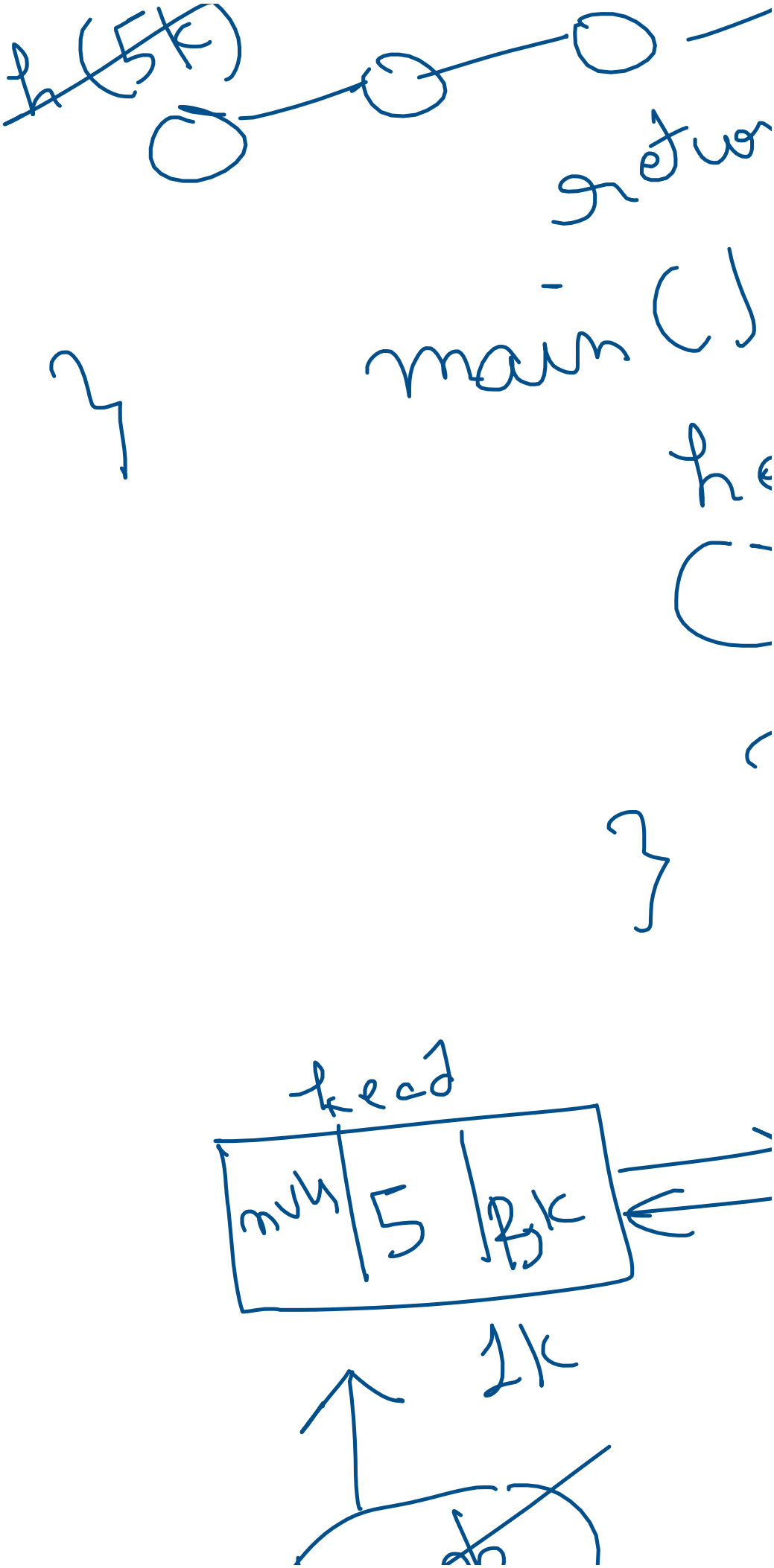
head.

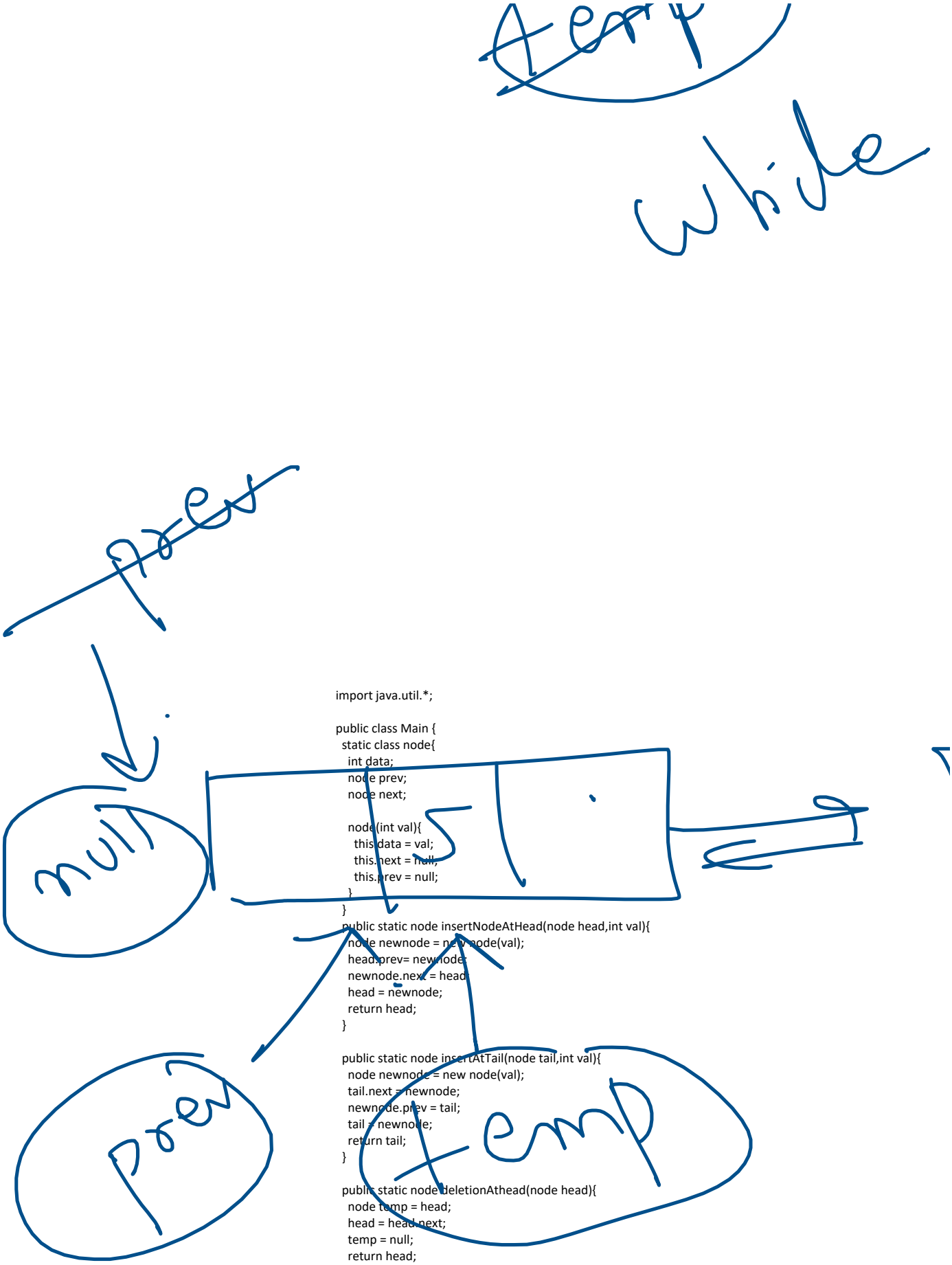






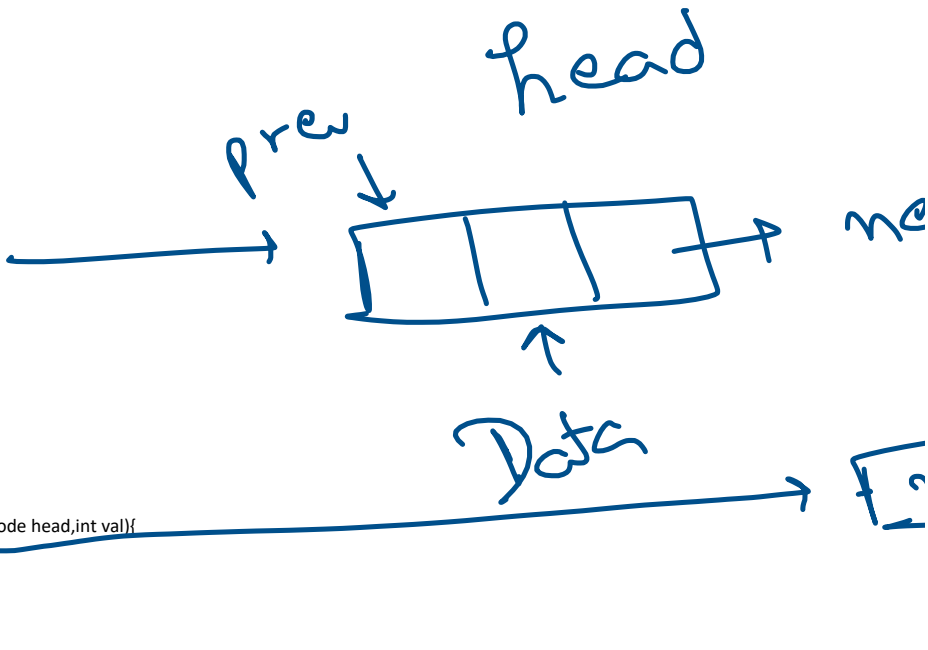
Deletion (5k) {


head
myy
5
ftk

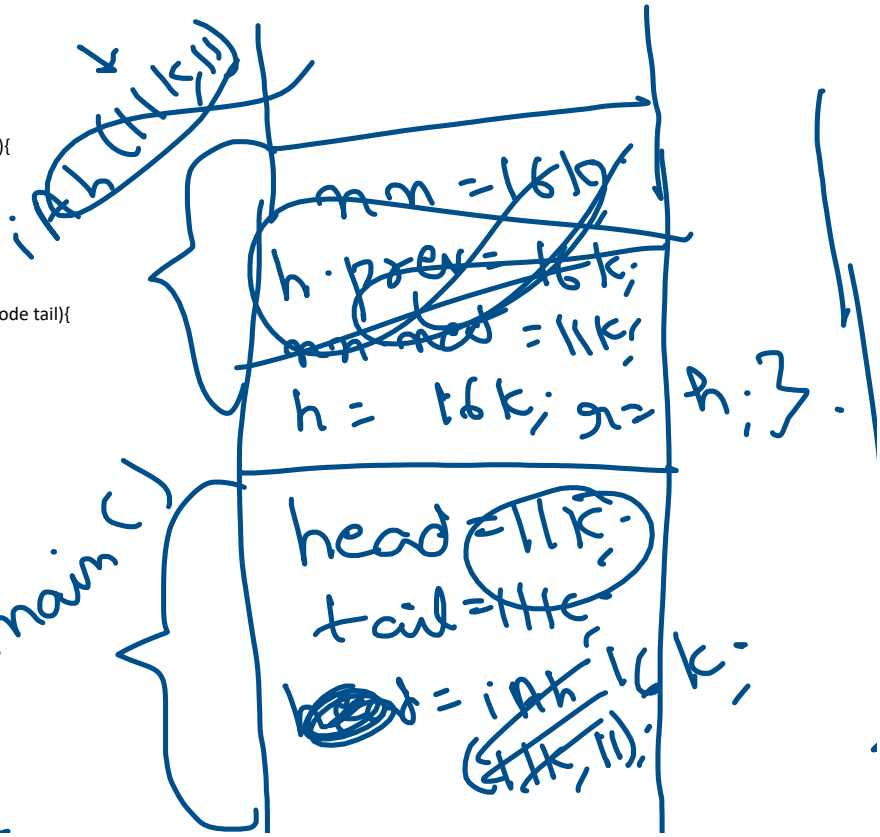


```
import java.util.*;

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

  node(int val){
   this.data = val;
   this.next = null;
   this.prev = null;
  }
 }
 public static node insertNodeAtHead(node head,int val){
  node newnode = new node(val);
  head.prev= newnode;
  newnode.next = head;
  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 node deletionAttail(node head,node tail){
  node prev = null;
  node temp = head;

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

```
    prev.next = null;
    tail = prev;
    return tail;
  }
  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 = insertNodeAtHead(head,11);
      head = insertNodeAtHead(head,10);
      head = insertNodeAtHead(head,9);
      head = insertNodeAtHead(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();

      tail = deletionAttail(head,tail);
      traverse(head);
      System.out.println();


    }
  }
```




```
import java.util.*;

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

   node(int val){
    this.data = val;
    this.next = null;
    this.prev = null;
   }
 }
 public static node insertNodeAtHead(node head,int val){
   node newnode = new node(val);
   head.prev= newnode;
   newnode.next =head;
   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 node deletionAttail(node head,node tail){
   node prev = null;
   node temp = head;

   while(temp.next != null){
    prev = temp;
    temp = temp.next;
   }
   prev.next = null;
   tail = prev;
   return tail;
  }
  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 = insertNodeAtHead(head,11);
        head = insertNodeAtHead(head,10);
        head = insertNodeAtHead(head,9);
        head = insertNodeAtHead(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();

        tail = deletionAttail(head,tail);
        traverse(head);
        System.out.println();


    }
}
```

Stack

Circular L.L

tail connect

no null;

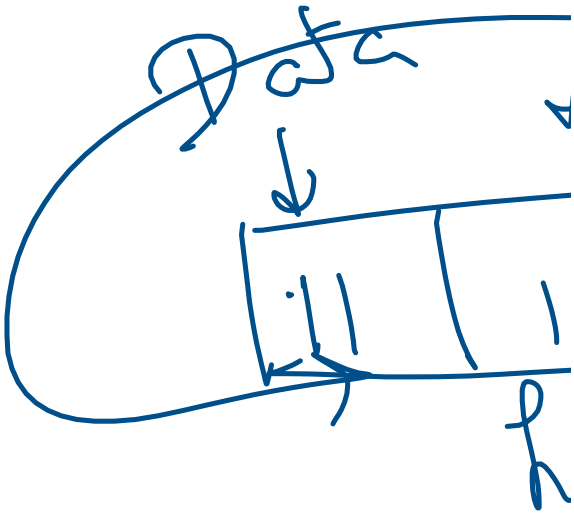
no st

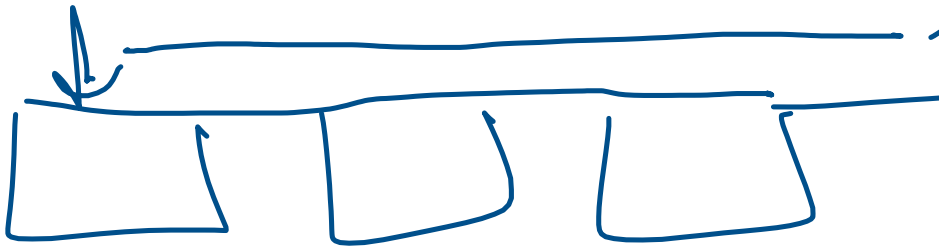
SinglyLL

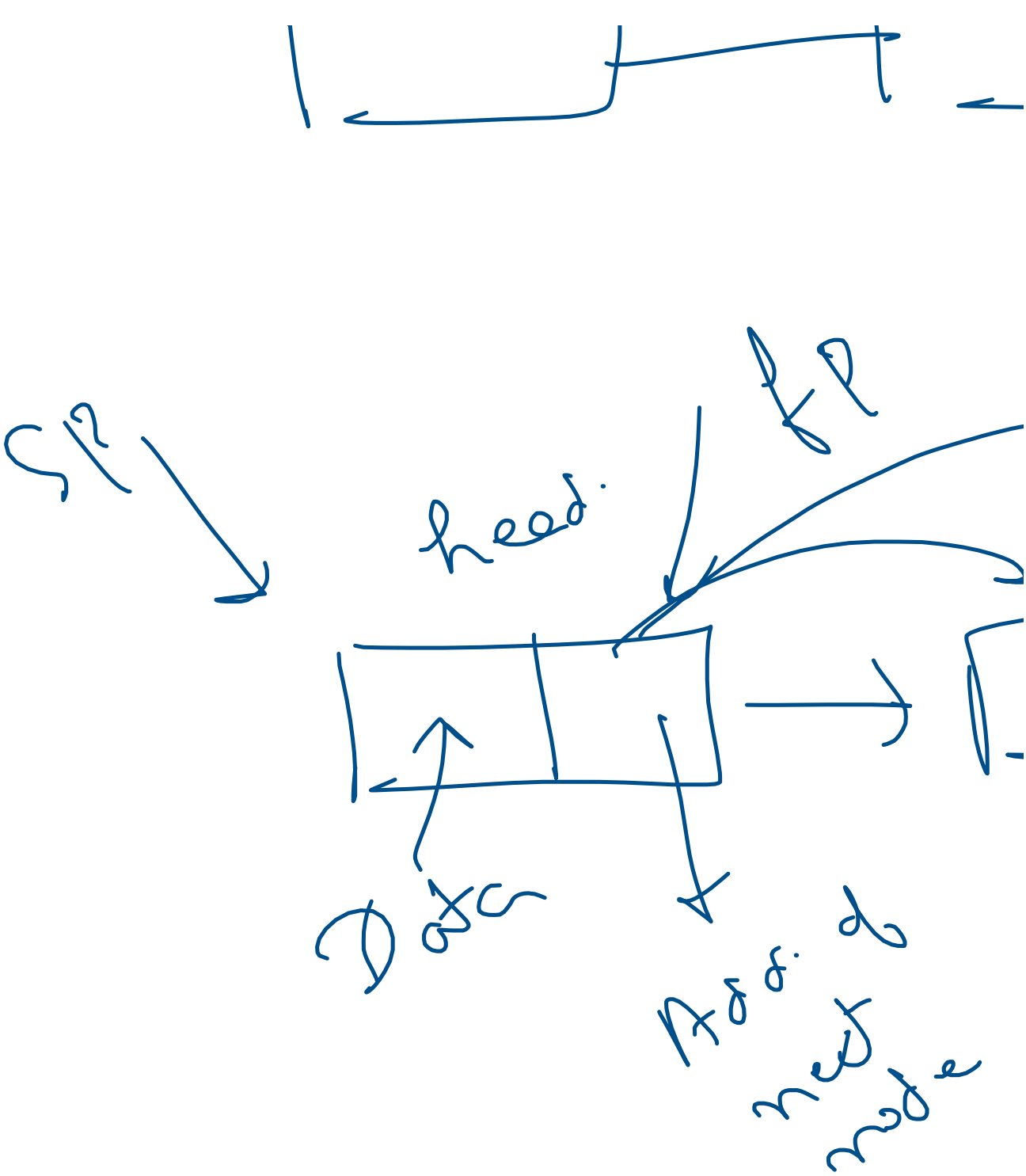
Data

Add

Circular






head.
fp
Data
Addr. of next node