Linked List

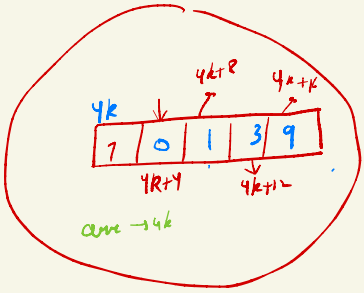
arr(0) → 4k arr → 4k

4k → 4k+4 →

int[] arr = new int[5];

arr → 4k 5*4

arr(0) 4
arr(1) → 4k+4



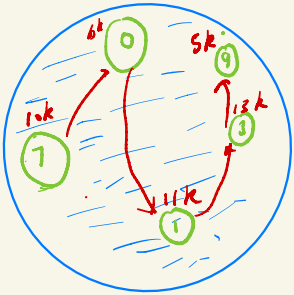
data
address of next Node object

20 bytes of autonomous space

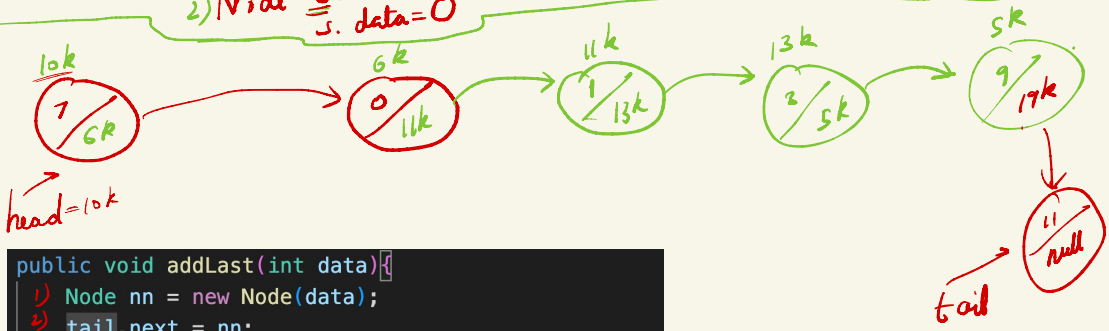
Student s

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

{ 1) Node f = new Node();
f.data = 7
2) Node s = new Node()
s.data = 0



s = 6k s = 6k




```java
public void addLast(int data){
    Node nn = new Node(data);
    tail.next = nn;

    tail = nn;
}
```

```java
public static void main(String[] args) {
    Node f = new Node(d: 7);
    Node s = new Node(d: 0);

    Node t = new Node(d: 1);

    // connecting first and second
    f.next = s;
    s.next = t;

    System.out.println(f.next.data); //0
}
```

head = 5k

head → 5k | 7 / 10k → 10k | 0 / 8k → 8k | 1 / 20k

break till 10:05

head = f
tail = t

```java
public void addLast(int data){
    Node nn = new Node(data);
    tail.next = nn;

    tail = nn;
}
```

⇒ 9

→ create new object

tail = 20k | 9 / null

⇒ 0 size

⇒ 1 size

⇒ x size list

```java
public class Main {
    static Node head;
    static Node tail;

    public static void addLast(int data){
        Node nn = new Node(data);

        if(head == null){ // in case of 0 size ll
            head = nn; // this is my first node of list
            tail = nn; // and also my last node of list
        } else {
            tail.next = nn; // attaching

            tail = nn; // shifting tail to last
        }
    }

    Run | Debug
    public static void main(String[] args) {
        head =null;
        tail = null;

        for(int i=1; i<=3; i++){
            addLast(i);
        }

        System.out.println(head.data);
        System.out.println(tail.data);
    }
}
```

0 size

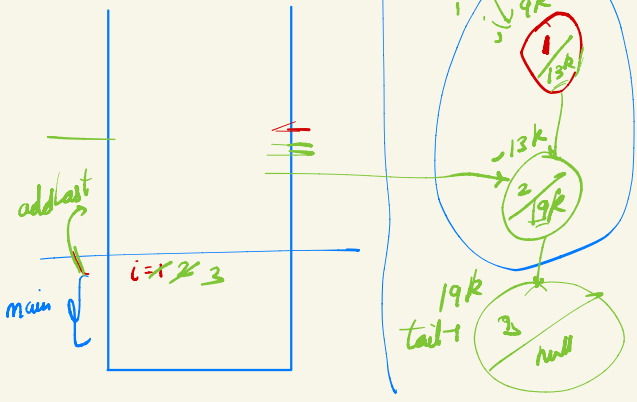
head = null

head = 9k

tail = 19k

i = 1

addlast (1)

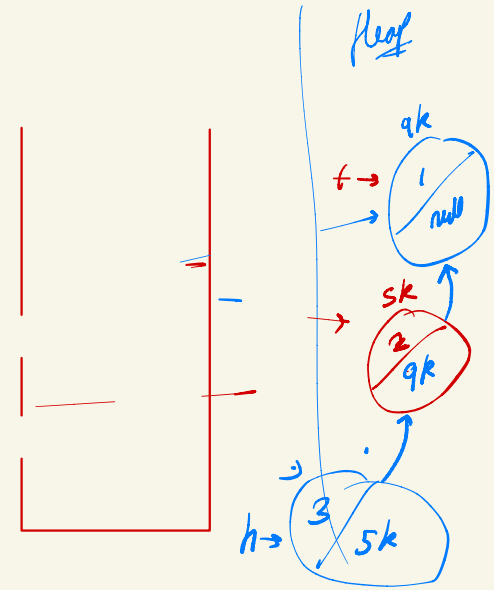


Ques Add a node at the start of LL.

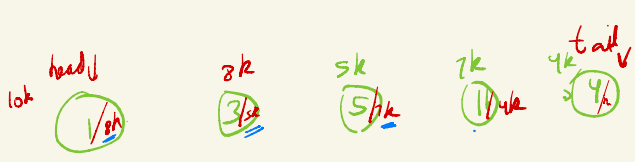
1) addFirst(1);
2) addFirst(2);
3) addFirst(3);

head = 5k
tail = 9k

```
public static void addFirst(int data){
    Node nn = new Node(data);

    if(head == null){
        head = nn;
        tail =nn;
    } else {
        nn.next = head;
        head = nn;
    }
}
```



display



curr = ~~10k~~ ~~8k~~ ~~5k~~ 7k 4k null

```
// homework
public static void display(){
    Node curr = head;

    while(curr!=null){
        System.out.print(curr.data+" ");
        curr=curr.next;
    }
}
```

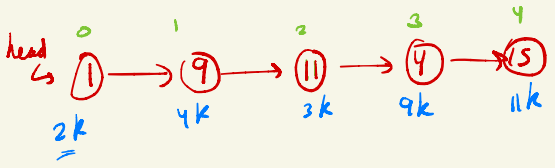
1 3 5 7 4

Ques Remove first element from linked list.

head = null

=> head = head.next;

Ques Get Node at particular index.



1) idx = -1
2) idx = 0
3) idx = 2
4) idx = 4

↳ idx = 6

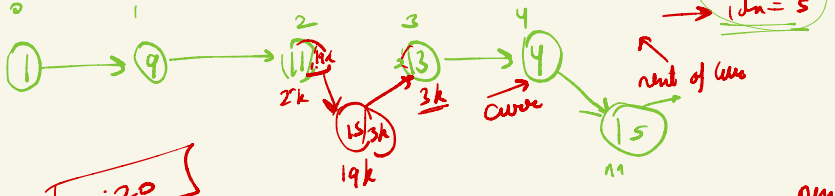
```java
public static Node getAt(int idx){
    if(idx < 0){
        System.out.println(x: "given index is out of range");
        return null;
    }

    Node curr = head;
    for(int i=0; i<idx; i++){
        if(curr==null){
            System.out.println(x: "given index is out of range");
            return null;
        }
        curr=curr.next;
    }

    return curr;
}
```

curr = head

i < 6

i = 0
i = 1
i = 2
i = 3
i = 4
i = 5

Ques (insert) add a Node at given index.



10:20

```java
public static void addAt(int idx, int data){
    Node nn = new Node(data);

    Node curr = getAt(idx-1);
    Node nextOfCurr = curr.next;

    curr.next= nn;
    nn.next = nextOfCurr;
}
```

idx = 0

rest of curr = null

nn.next = rest of curr

2k.next = 19

NOC = 3k

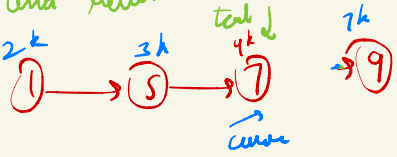
19k.next

i <= idx-1

i = 1

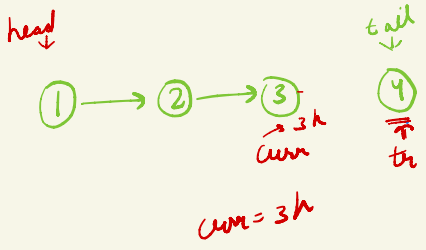
Ques Remove and return the last Node of ll.

⇒



curr = 2k 3k 4k

```java
public static Node removeLast(){
    if(head==null){ // 0 size
        return null;
    } else if(head==tail){ // 1 size
        Node tr=tail;
        removeFirst();
        return tr;
    } else {
        Node curr = head;
        while(curr.next!=tail){
            curr = curr.next;
        }

        curr.next=null;
        // updating tail
        Node tr = tail;
        tail = curr;

        return tr;
    }
}
```



curr = 3h

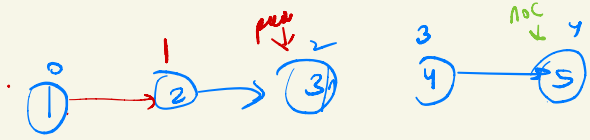
4 is deleted from ll

but not from heap.

→ write on your own code & test

→ dry run the code

Ques Remove at given index.

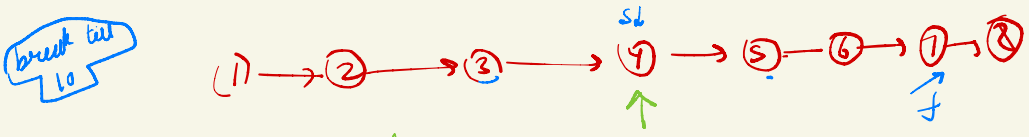



```java
public static void removeAt(int idx){
    int size = getLength();
    if(idx==0){
        removeFirst();
    } else if(idx==size-1){
        removeLast();
    } else {
        Node prev = getAt(idx-1);
        prev.next = prev.next.next;

        // Node nextOfCurr = prev.next.next;
        // prev.next = null;
        // prev.next = nextOfCurr;
    }
}
```

```java
public static int getLength(){
    Node curr = head;
    int len =0;

    while(curr!=null){
        curr = curr.next;
        len++;
    }

    return len;
}
```

1) These were the building blocks.
2) We have a default linked list.

Ques Find middle of linked list.



Ques what if we wanted to return first middle

fast.next!=null && fast.next.next!=null

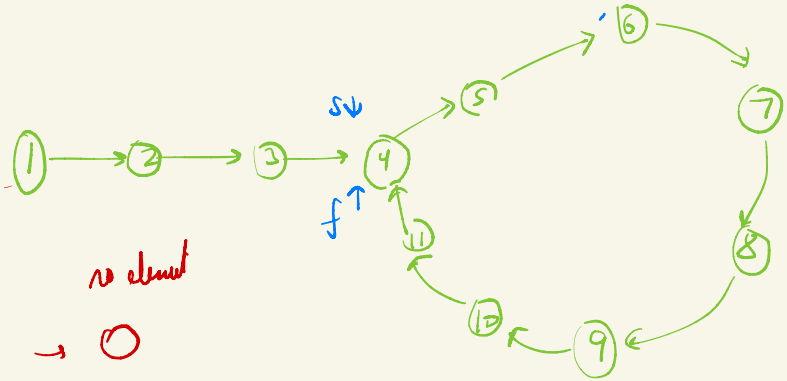
```java
public ListNode middleNode(ListNode head) {
    ListNode slow = head;
    ListNode fast = head;

    while(fast!=null && fast.next!=null){
        slow = slow.next;
        fast = fast.next.next;
    }

    return slow;
}
```

even odd

slow → 1 node per move

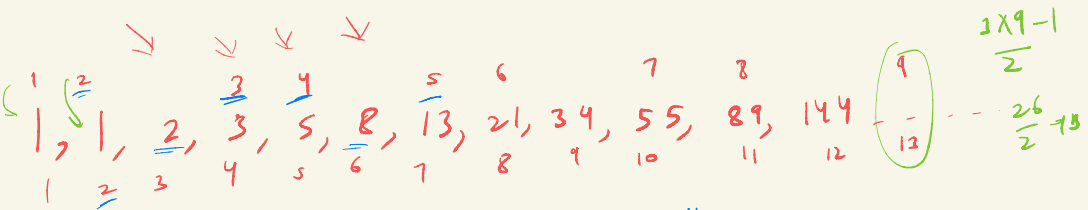
fast → 2 nodes per move

hare and tortoise method



no cloud

1) Check if there is a cycle
2) Keep fast at the intersecting node and take slow back to head
3) Move slow and fast at equal pace.
4) They will coincide again → this is the starting point.



1st odd → 1st fib

2nd odd → 2nd fib

3rd odd → 4th fib

4th odd → 5th fib

5th odd → 7th fib

6th odd → 8th fib

$\frac{3x-1}{2}$

xth odd ↓ $\left(\frac{3x-1}{2}\right)^{th}$ fib

odd	fib
3	4
5	7
6	8
7	10

3 even element

9 → 13

10 → 14

$x_1 \Rightarrow \left(\frac{3x-1}{2}\right)$

Find xth fib

1 2 3 4 5 6 7

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 - - - -

Faith

Expectation

My

```
int lt = fun(x-1);
int ltt = fun(x-2);

int ans = lt + ltt;
```

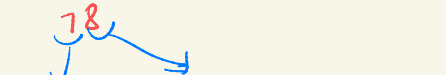
f(x-1) ✓

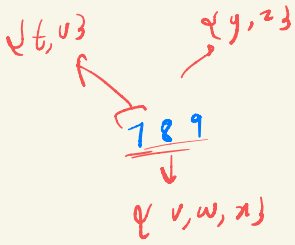
f(x-2) ✓

5th odd fib number → $\frac{3\times5-1}{2}$ → 7th

dues subsequence question

0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz

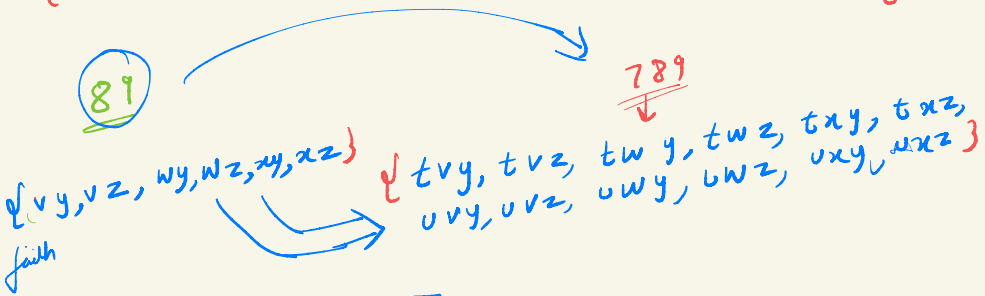




tvy, tvz, twy, twz
txy, txz
uvy, uvz, uwy, uwz
uxy, uxz

Faith
Return all the combination for "8"
{v, w, x}

Expectation
Return all the combinations when I will send the string
78 → {tv, tw, tx, uv, uw, ux}

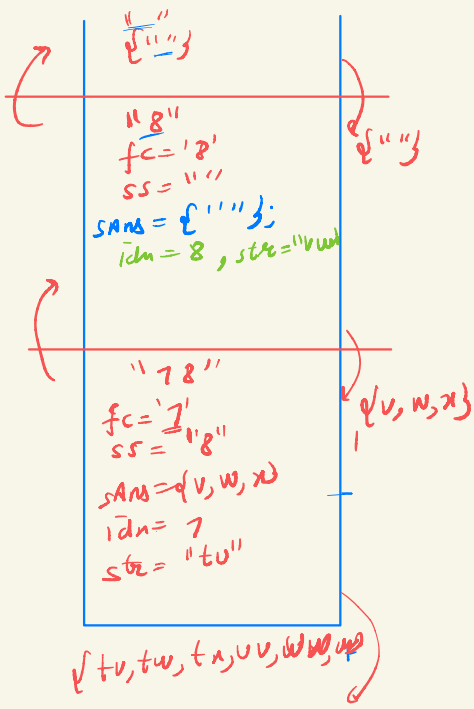


7
↓
tu

8 → vwx
9 → yz

why we should not return empty ds ?

```java
public ArrayList<String> getKPC(String ques){

    char first_char = ques.charAt(0);
    String ss = ques.substring(1);
1)  ArrayList<String> sans = getKPC(ss);

2)  int idx = first_char - '0';

3)  String str = sarr[idx]; // tu

4)  ArrayList<String> ans = new ArrayList<>();
5)  for(int i=0; i<str.length(); i++){
        char ch = str.charAt(i); // t;
        for(int j=0; j<sans.size(); j++){
            String sansString = sans.get(j);
            String myAnsString = ch + sansString;
                                 u + x
            ans.add(myAnsString);
        }
    }

    return ans;
}
```



int idx = '8' - '0'

str = "tu"

sAns = {"v", "w", "x"}

ans = {tv, tw, tx, uv, uw, ux}