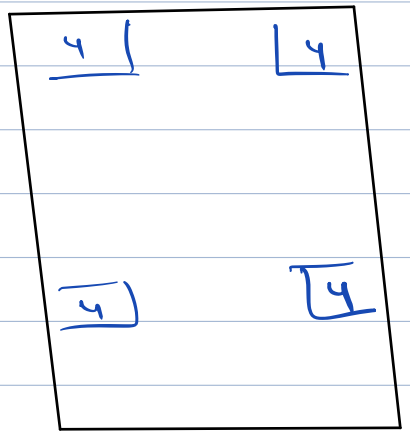
Today's Content

→ Intro to LL
→ Create LL
→ Print LL
→ Insertion
→ Print Reverse LL.

RAM

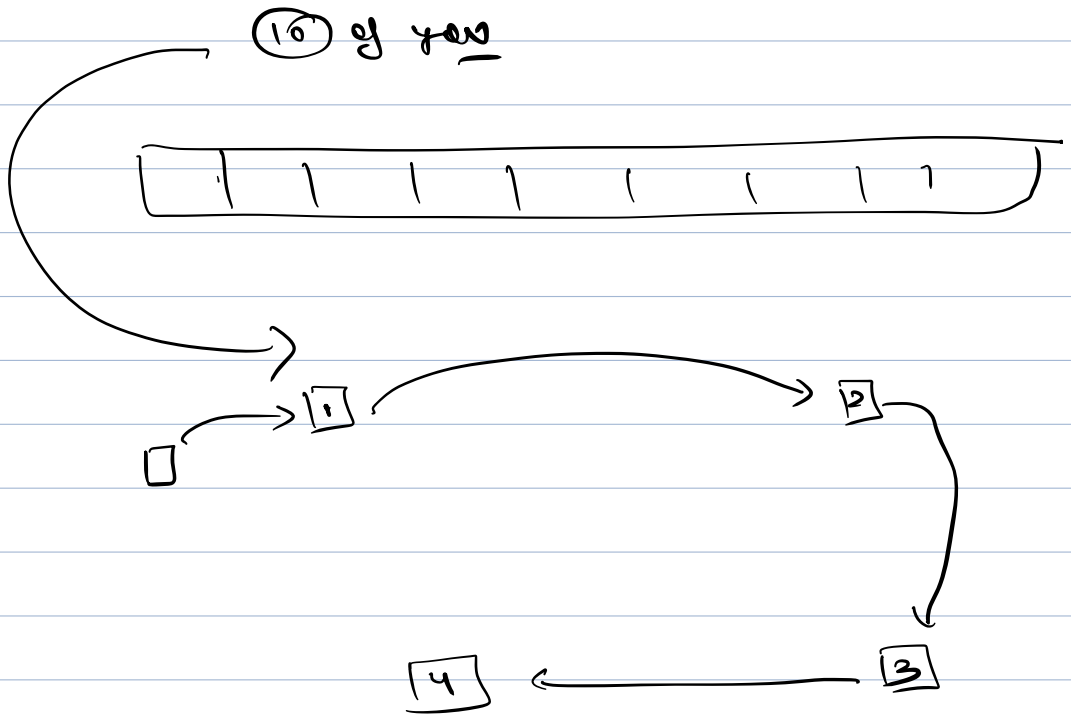


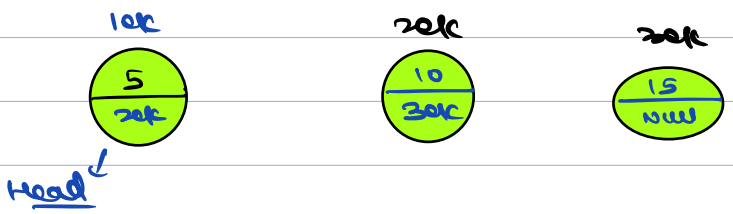
int [] arr = new int[10];

LinkedList

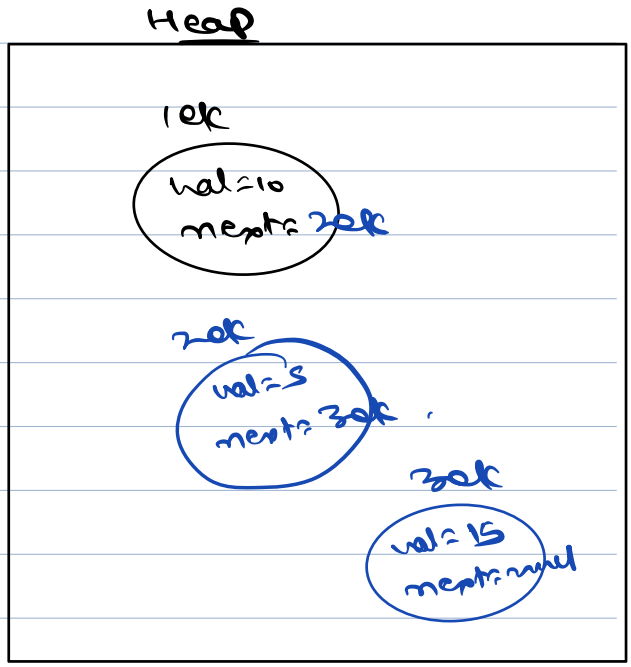
↓

doesn't require contiguous memory.





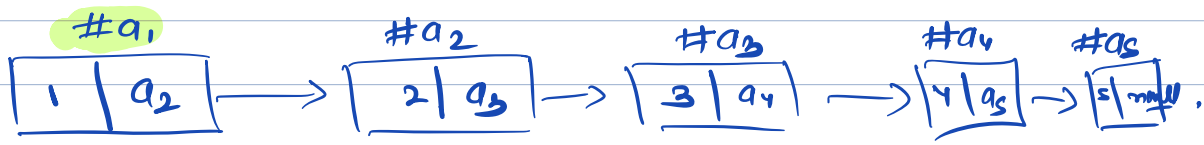
```
class Node {
    int val;
    Node next;
    Node(x) {
        val = x
        next = null
    }
}
```




```
Node head = new Node(10);
Node t = new Node(5);
head.next = t;
t = new Node(15);
head.next.next = t;
```

Ques) Create a LL with n nodes, with data 1-n, & return head node.

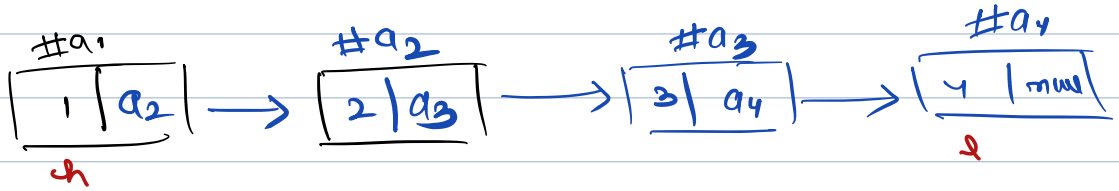
e.g., N = 5




```
class Node {
    int val;
    Node next;
    Node(x) {
        val = x
        next = null
    }
}
```

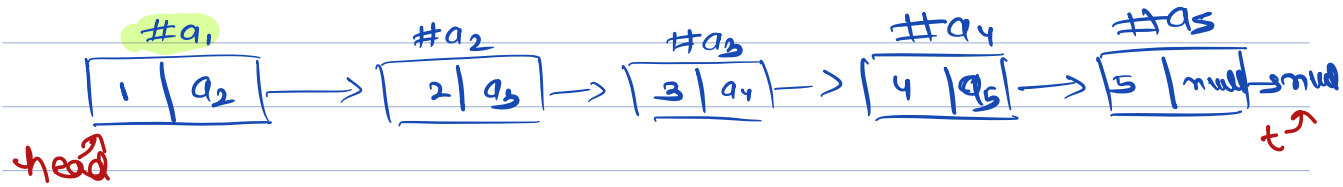
```
Node createList(int n) {
    Node h = new Node(1);
    Node l = h;
    for (i = 2; i <= n; i++) {
        l.next = new Node(i);
        l = l.next;
    }
    return h
}
```

h = a1, l = a4



Ques) Given head node, return size.

e.g.,



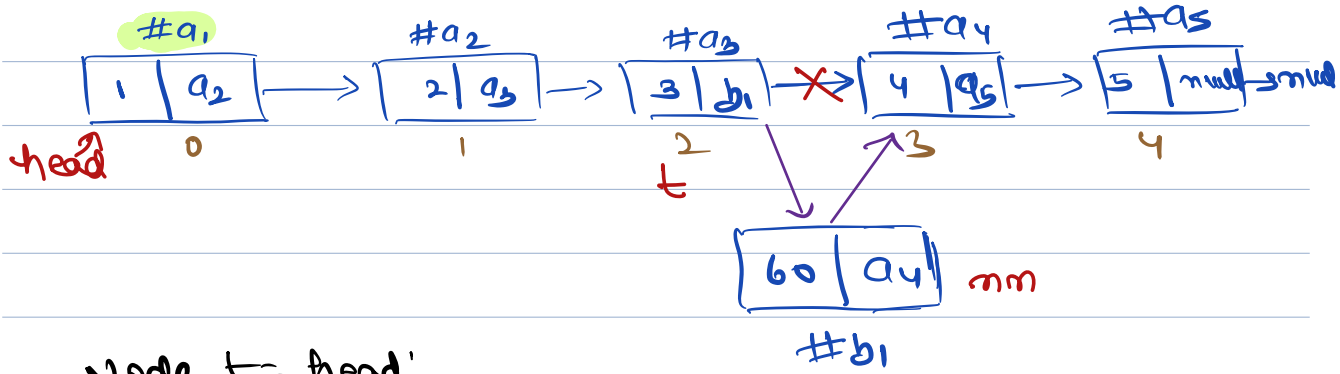

```
class Node {
    int val;
    Node next;
    Node(x) {
        val = x
        next = null
    }
}
```

```
int sizeOfLL(Node head){
    Node t = head;
    int c = 0;
    while(t != null){
        c++;
        t = t.next
    }
    return c
}
```

Break 10:00pm - 10:12pm

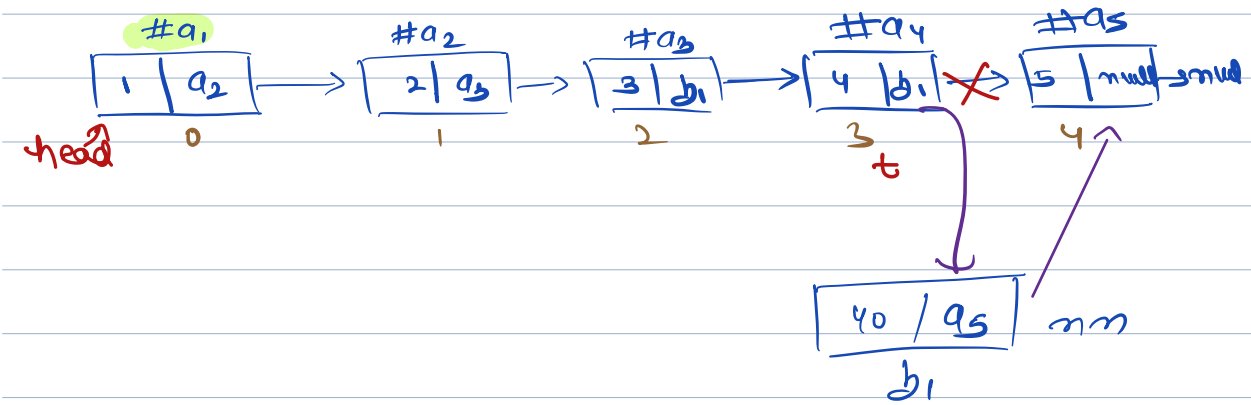
Q) Given a LL, insert a new node with data v at index p.

Ex1 :- v = 60, p = 3

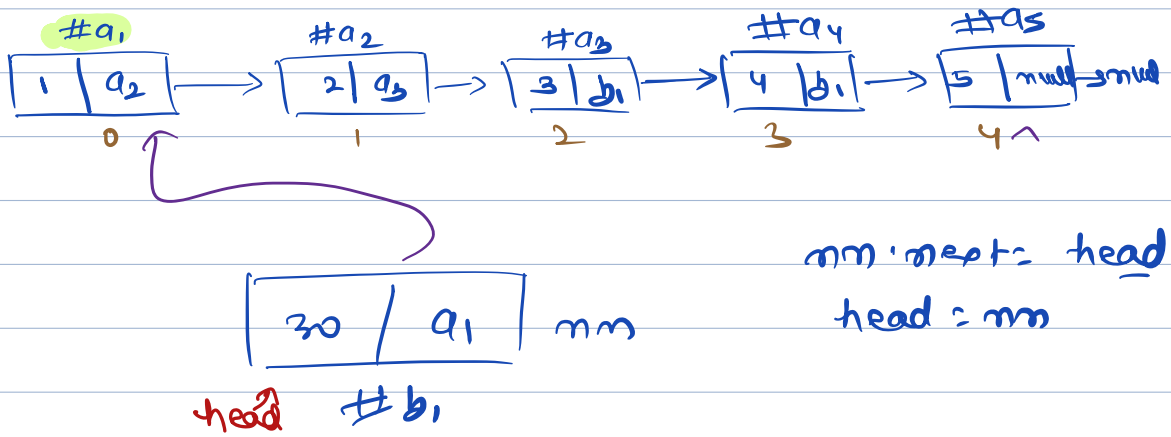


Node t = head;
move t p-1 times
// node exists
nn.next = t.next
t.next = nn;

e.g 2) :- v = 40, p = 4



e.g 2) :- v = 30, p = 0



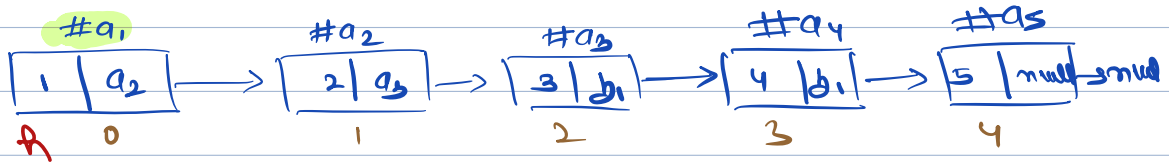

```
Node insert (Node h, int v, int p) {
    Node n = new Node(v);
    if (p==0) {
        nn.next = head;
        head = nn
        return head
    }

    Node t = head;
    // move t p-1 times
    for (i=1; i<p; i++) {
        t = t.next
    }
    nn.next = t.next
    t.next = nn;
```

return head;

}

Ques) given linkedList print reverse.

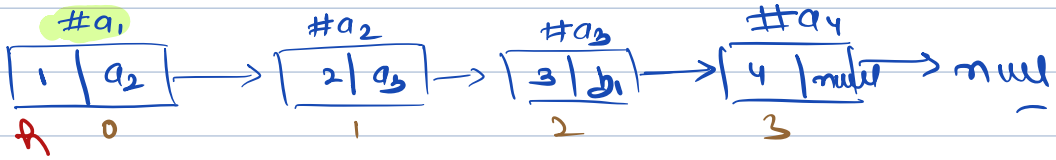


// Recursion :-

```
void printRev(Node h) {
    if (h == null) { return; }

    printRev(h.next);
    print(h.val);

}
```



4, 3, 2, 1

```
void printRev(Node h=a1) {
    if (h==null) { return; }
    printRev(h.next);
    print(h.val);
}
```

3

```
void printRev(Node h=a2) {
    if (h==null) { return; }
    printRev(h.next);
    print(h.val);
}
```

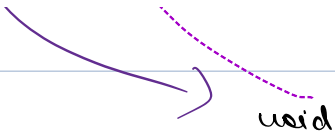
3

```
void printRev(Node h=a3) {
    if (h==null) { return; }
    printRev(h.next);
    print(h.val);
}
```

3

```
void printRev(Node h=a4) {
    if (h==null) { return; }
    printRev(h.next);
    print(h.val);
}
```

3



```
void printlev(Node h = null ){
  if (h= null) { return }
printlev(h.next)
print(h.val);

}
```