Todays Content

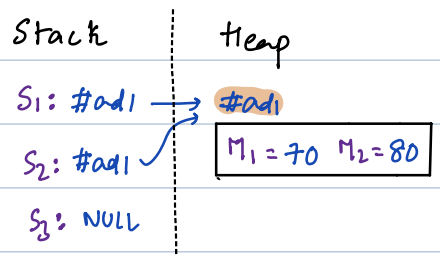
a) Linked list Basics

b) Given Head Node

1. Print size
2. Print linked list
3. Insert a new Node in linked list
4. Delete a node in linked list
5. Reverse a linked list
6. Return middle of a linked list

class & Object basics

```
class student{
    int M1, M2; → Attributes
    student(int c1, int c2){ → Constructor: Used to Initialize Attributes
    }  M1 = c1 ; M2 = c2
}
void main(){
    Student S1 = new Student(50, 60);
    Objectref   Used to Create object
    Store add
    S1.m1 = 70; ✓
    print(S1) : add1
    Student S2 = S1; ✓
    S2.M2 = 80; ✓
    Student S3 = NULL ✓ Note: Storing NULL as address is valid
    S3.M1 = 90; // Null Pointer Exception:
                We are trying to access data at NULL.
}
```



Linked List Basics

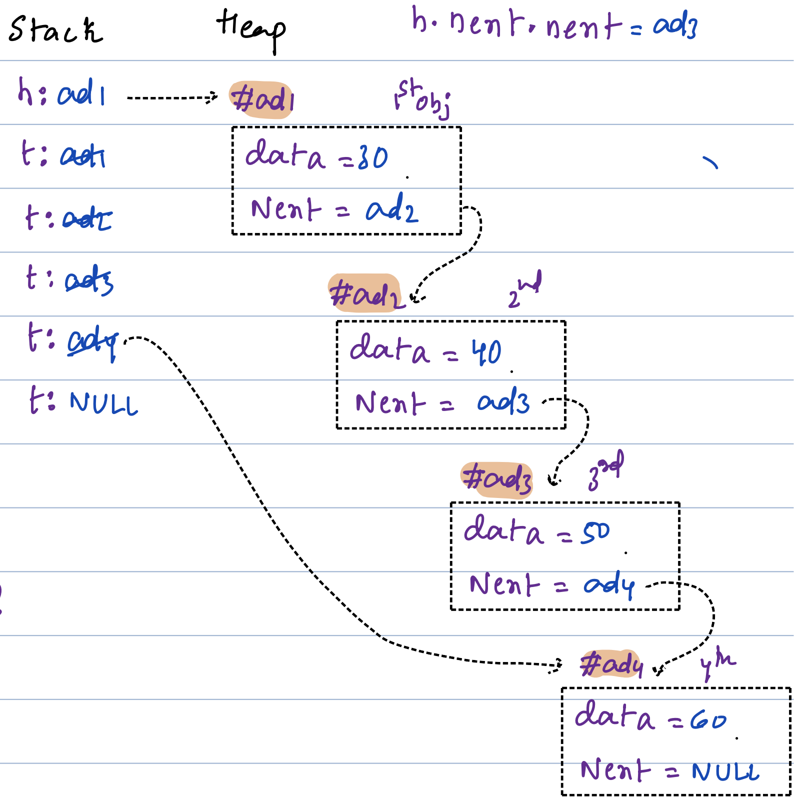
```
class Node{
    int data;        → variable
    Node next;       → object reference : Can hold address of Node Object
    Node(int n){     → constructor, In python slightly diff.
        data = n
        next = NULL
    }
}
```

```
void main(){
    Node h = new Node(30)
    (objref)  (object creation)
```

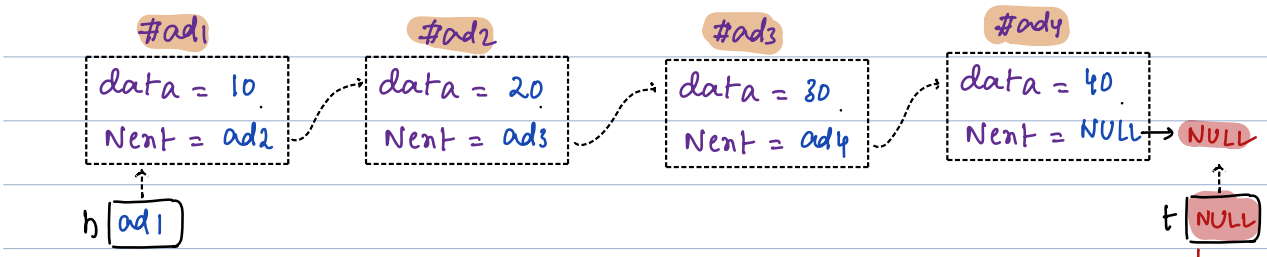
h.next.data = 40

h.next.next = ad3

```
    print(h)          : ad1
    print(h.data)     : 30
    print(h.next)     : NULL
    Node t = h ✓
    t.next = new Node(40) ✓
    t = t.next ✓
    t.next = new Node(50) ✓
    t = t.next ✓
    t.next = new Node(60) ?
    t = t.next
    // Printing List
    t = h;
    while(t != NULL){      Note: NULL is used to indicate end of traversal.
        print(t.data)  // 30 40 50 60
        t = t.next
    }
    print(h.data) : 30
    print(h.next.data) : 40
    print(h.next.next) :
}
```



1@) Given a head Node of a linked list, return size of it.



Node t = h

c=0 : c=c+1 t=t.next | c=c+1 t=t.next | c=c+1 t=t.next | c=c+1 t=t.next | Stop.

```
int size(Node h){
    Node t = h;
    int c = 0;
    while(t != NULL){
        c = c+1;
        t = t.next
    }
    return c;
}
```

TC: O(N)

SC: O(1)

Note: Edge Cases

Say we want to access h.data / h.next?

if h != NULL:

Access h.data / h.next

Say we access h.next.data?

if h != NULL && h.next != NULL

Access h.next.data

En: h = NULL : Empty Linked List, Is it working? Working.

Issue in Below Code:

```
Node t = h;        ---> h = NULL, Empty List
int c = 1;
while(t.next != NULL){      NULL.next = Null Pointer Exception.
    c = c+1
    t = t.next
}
return c;
```

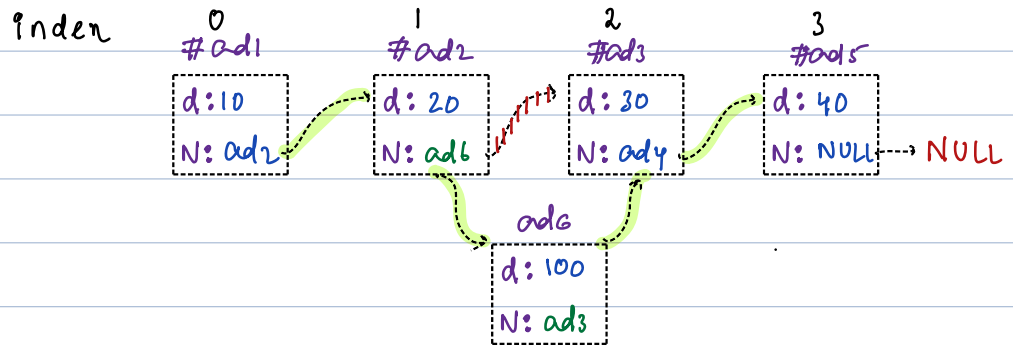
int a = null

Insert in a linked list

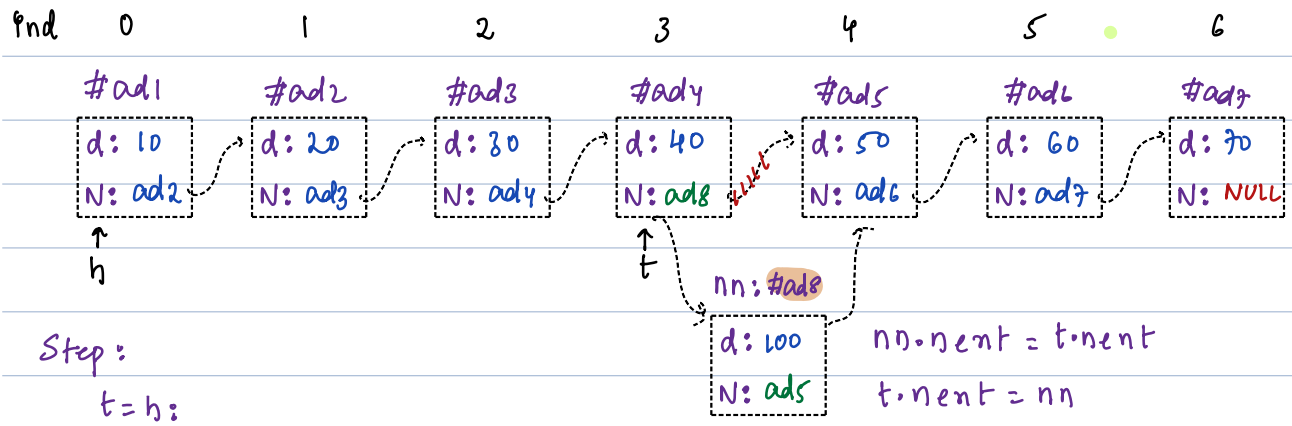
Given a head node of linked list & B, C;
Create a node with value of B and insert at pos C
Return head Node of linked list after Insertion.

Note: Assume pos is 0-based Indexing
0-based means : 1st Node Index at 0

Ex: B = 100, C = 2, Node nn = new Node(100)



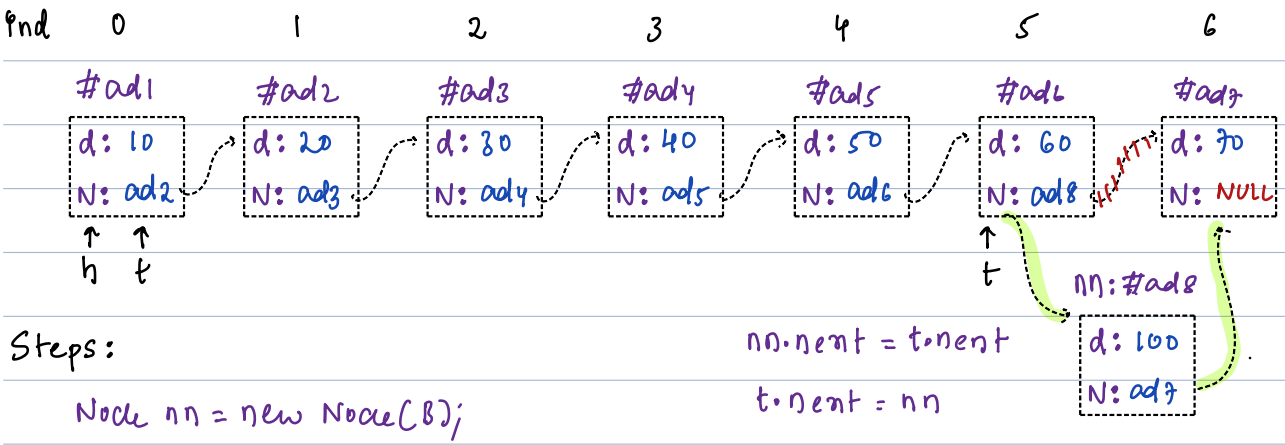
Ex2: B = 100, C = 4 Node nn = new Node(B);



Step:
t = h:
updated t = t.next 3 times.

nn.next = t.next
t.next = nn

Eg2: B = 100, C = 6 Node nn = new Node(100);



Steps:

Node nn = new Node(B);

Node t = h;

Perform t = t.next, C-1 times

nn.next = t.next

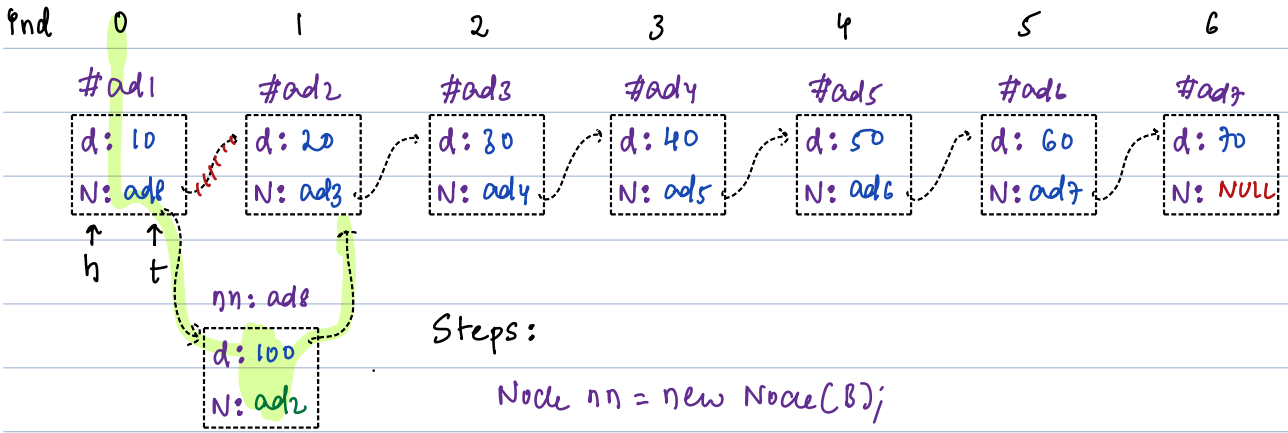
t.next = nn

Insert Node:

nn.next = t.next

t.next = nn

Eg4: B = 100, C = 1 Node nn =



Steps:

Node nn = new Node(B);

Node t = h;

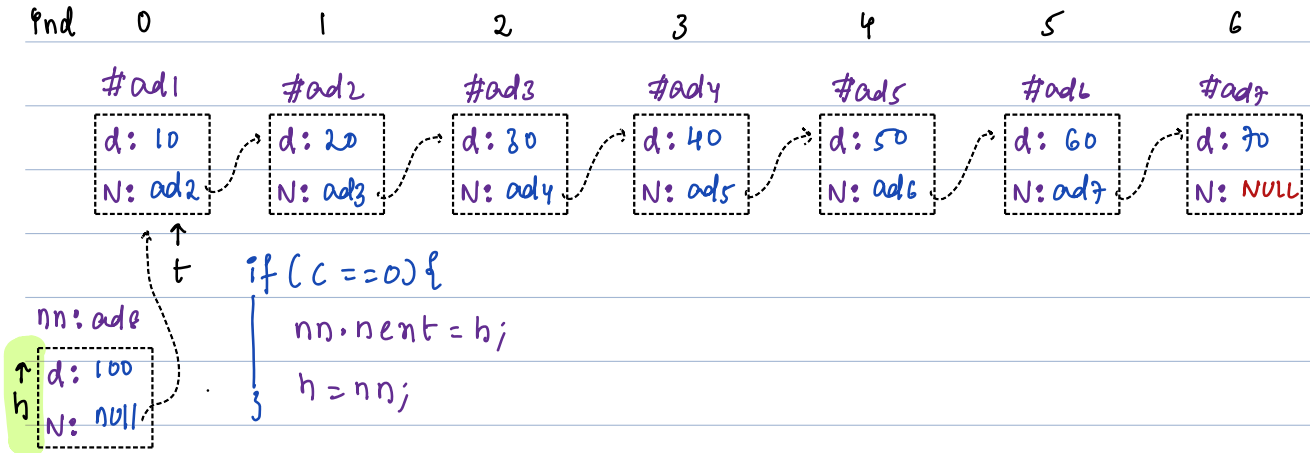
Perform t = t.next, C-1 times

Insert Node:

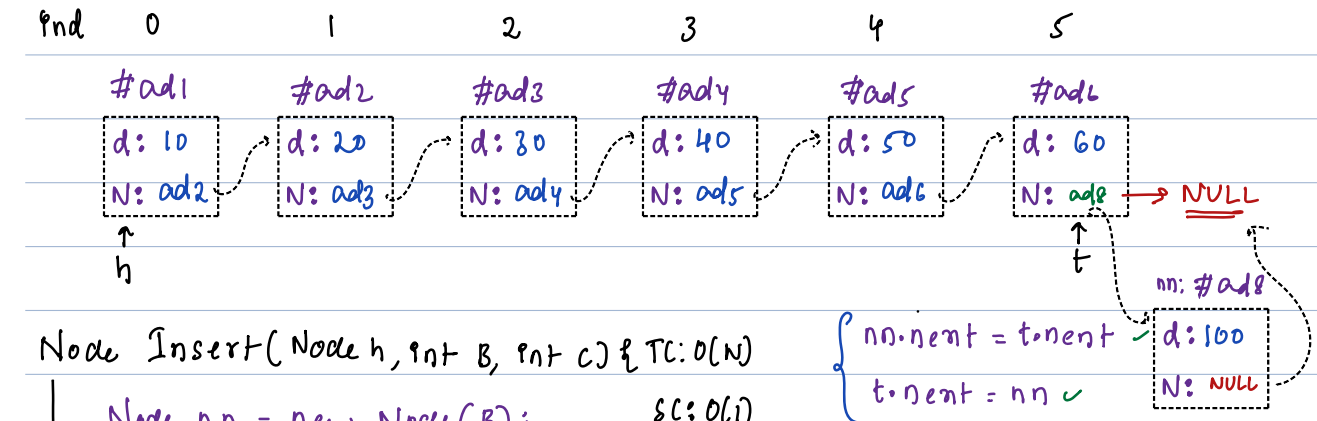
nn.next = t.next

t.next = nn

Eg2: B=100, C=0 : Edge Case:



Eg2: B=100, C=8 : If position not available insert at last



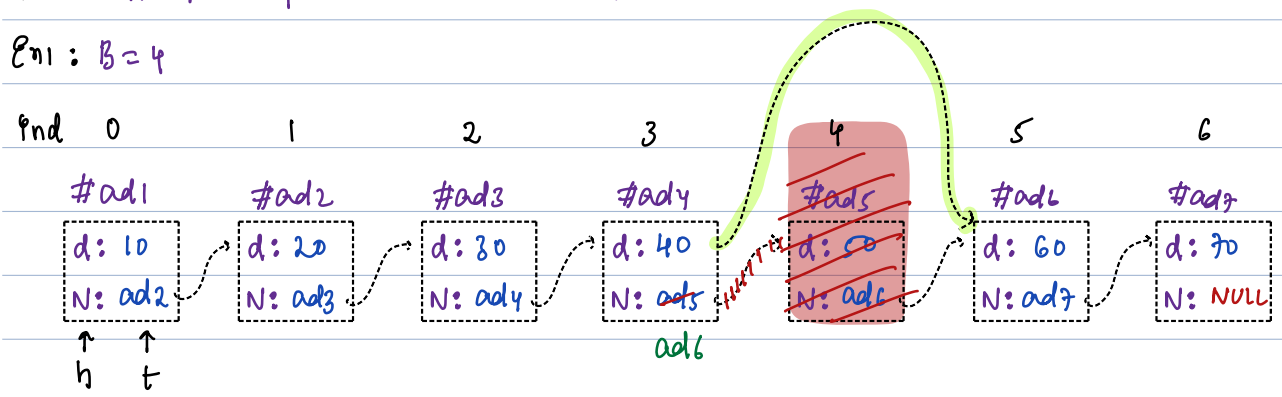

```
Node Insert(Node h, int B, int C) { TC: O(N)
    Node nn = new Node(B);                SC: O(1)
    if(C==0 || h==NULL) {                 } Edge Case
        nn.nent=h; h=nn;
        return h;
    }
    Node t=h;
    for(int i=1; i<=C-1; i++) {// Iterate C-1 times.
        if(t.nent==null) { break}
    }   t=t.nent;
    // Insert node
        nn.nent = t.nent
        t.nent = nn
        return h;
}
```

Delete: Given h node of linked list & B.

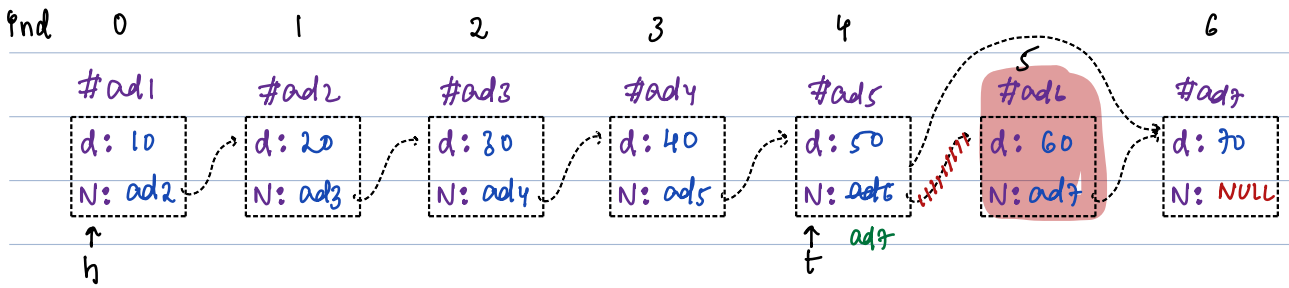
Delink node at index pos = B.

Note: Given B is present in linked list

Ex1 : B = 4



Ex2: B = 5



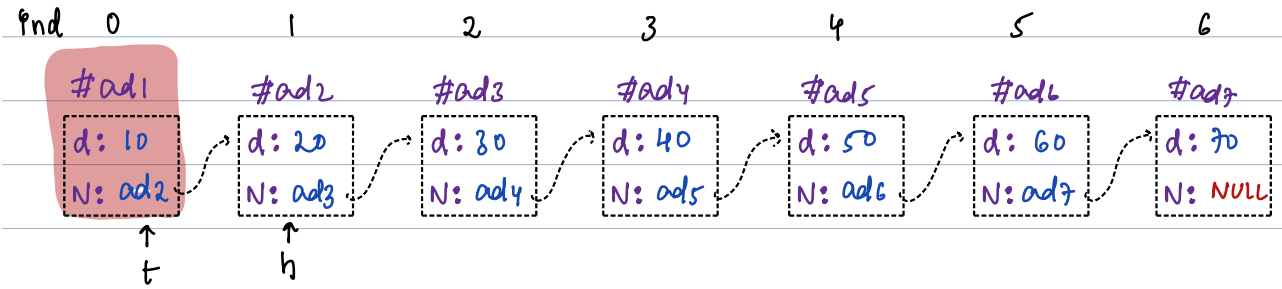
Steps: Node t = h;

update t = t.next B-1 times

t.next = t.next.next

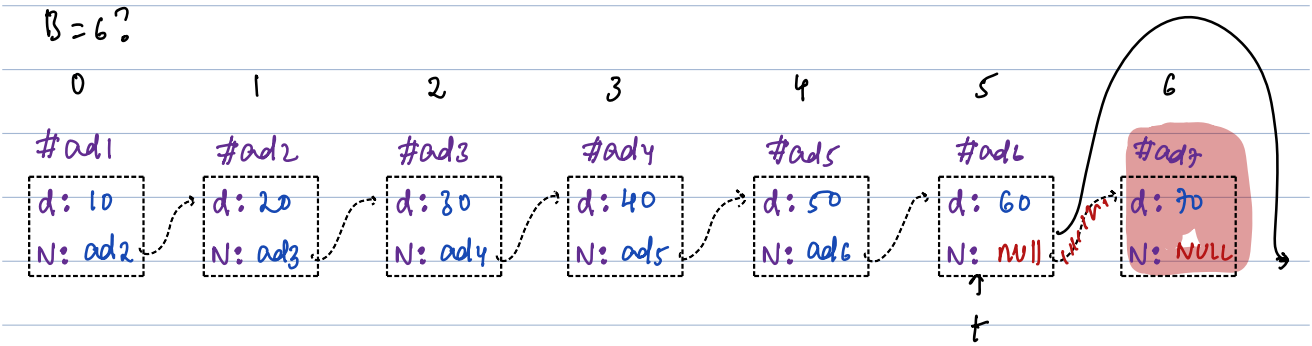
t.next = t.next.next

Ex3: B = 0

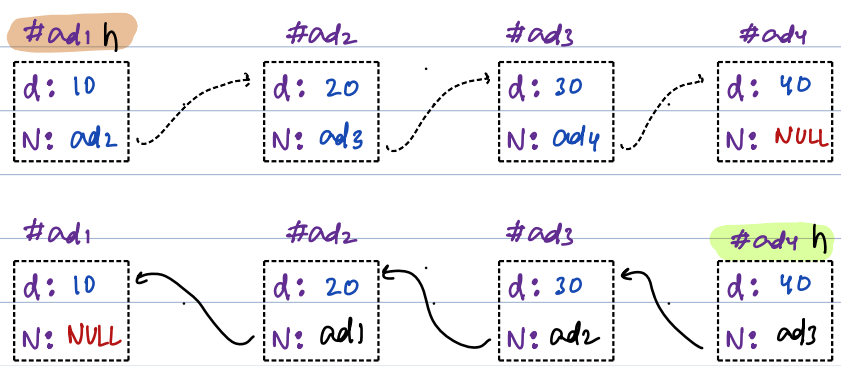



```
if (B==0){
    h = h.next
    return h;
}
```

```
Node Delete(Node h, int B){ TC: O(N) SC: O(1)
    if(h==null){return h}
    if(B==0){
        h=h.next;
    }return h;
    // Iterate B-1 times
    Node t=h;
    for(int i=1; i<=B-1; i++){
        t=t.next;
    }
    t.next=t.next.next
    return h;
}
```



Q: Reverse given LinkedList & return head Node.



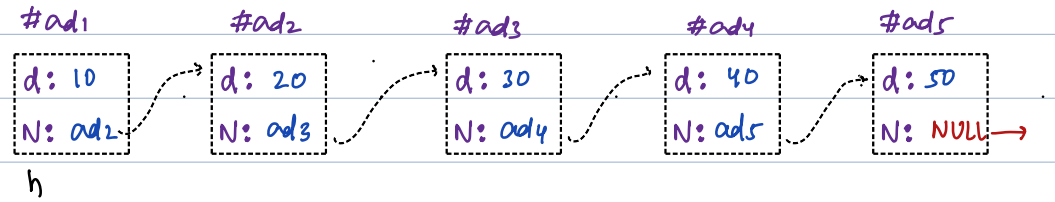
Q3: Reverse linkedlist

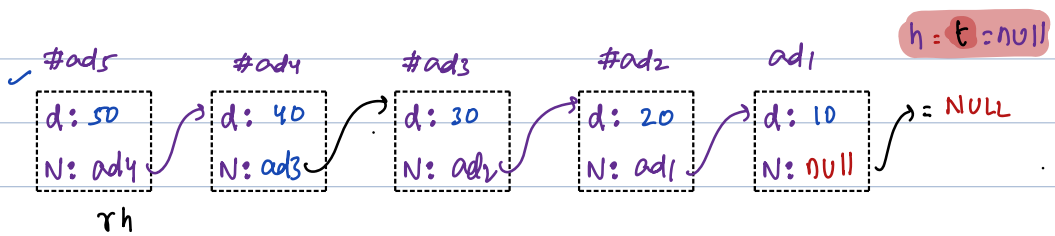
Given a linkedlist, reverse entire linked list & return head node

We can only change ref of a node.

Note: No Extra Space & We cannot change val of node.

Eg1:






```
Node reverse(Node h){ TC:O(N) SC:O(1)
    Node t=h, rh=null;
    while(h!=NULL){
        t=t.next; } Isolate head node
        h.next=null; } → skip it
        h.next=rh; } Inserting h at start
        rh=h; } updating rh
        h=t.
    }
    return rh
}
```

Note: Inserting nodes at start can get you reversed