Todays Content:

a) Class & Object Concept → Oops * → Design
b) Object & Object reference
c) Object reference as memember
d) linked list Basics
e) Problems in linkedlist

Class : It is a blue print

 → Blueprint

Object : Real Creation of blue print

 → Actual house

class Car {

Attributes of class Car
- → company
- → Colour
- → Mileage

Functions of class Car
- → drive()
- → ACC()
- → Lights()

}

→ Car : Mehta {Object}

company : Toyota
Colour : Red
Mileage : 15 km/1 litre

drive()
ACC()
Lights()

→ Car : Bharat {Object}

company : Ford
Colour : Silver
Mileage : 25 km/ltr

drive()
ACC()
Lights()

Student Class :

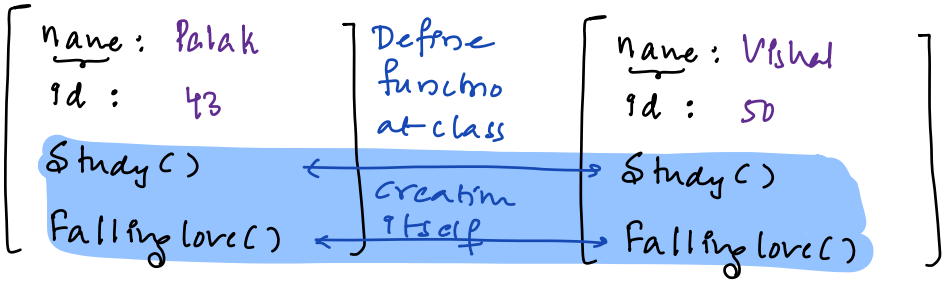
1) Creation of Class Syntax, In your language of Choice

2) Object Creation,

```
ClassName var = new ClassName();
```

```
Class Student {
    String name
    int    id
    Study()          // functionalities define at Class
    FallingLove()    // Creation itself
}
```

```
Student S1 = new Student()        Student S2 = new Student()
```




```
S1.name = "Palak"
S1.id   = 43
S1.Study()
S1.FallingLove()

S2.name = "Vishal"
S2.id   = 50
```

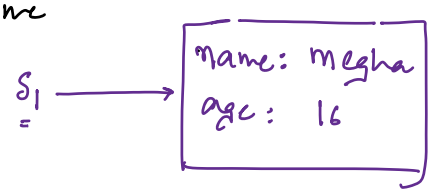
Eg:

```
class Student
{
    String name
    int age
    Student() {
    }
}
```

Object reference — Will create object

```
Student S1 = new Student()
```

ClassName



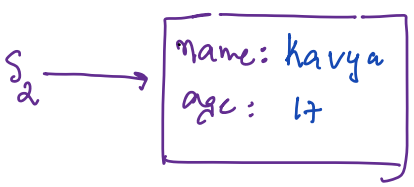
$S_1.name$ = megha $S_1.age$ = 16

```
Student S2; // S2 is object reference of Student Class
```

$S_2.name$ = "kavya" : Error

Null pointer Exception

```
S2 = new Student()
```

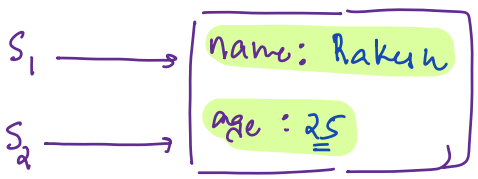


// Object Refernce:

Classname s; // Only object refernce created

```
class Student
    String name
    int age
    Study() {
    }
}
```

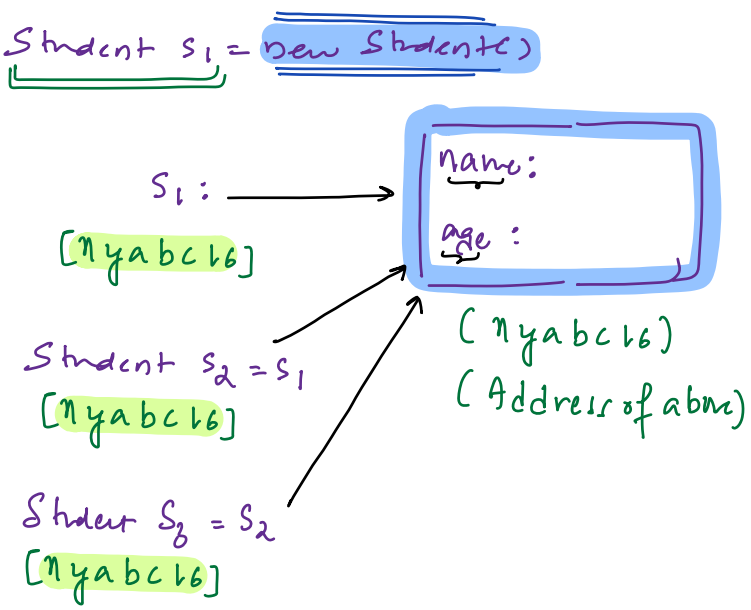
```
Student s1 = new Student()
```




```
s1.name = Rakesh   s1.age = 24
Student s2 = s1
s2.age += 1
print(s1.age) // 25
```

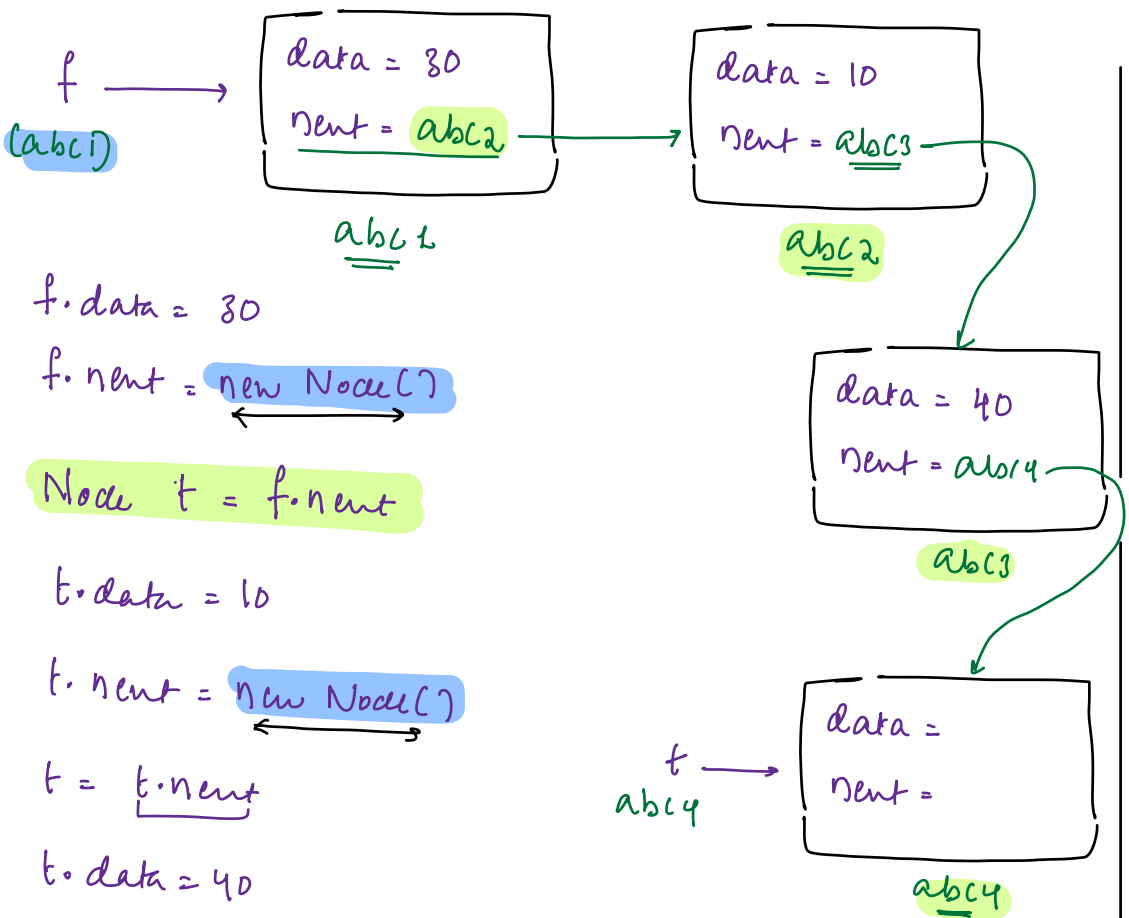
// Conceptual: Multiple Object reference can point same Object

```
Student s1 = new Student()
```



```
Class Node {
    int data // It's just int
    Node next // Object reference of Class Node, can hold node Object
}
```

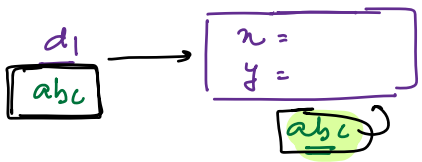
```
Node f = new Node();
```




```
f.data = 30
f.next = new Node()

Node t = f.next

t.data = 10
t.next = new Node()

t = t.next

t.data = 40
t.next = new Node()

t = t.next
    abc4
```

```
Class Node {
    int x, int y
}
```

```
Node d1 = new Node();
```



$d_1.x = 10$

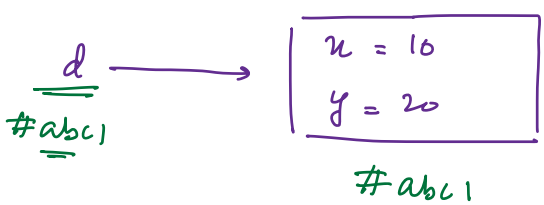
$d_1.y = 20$

→ Initialization of Attributes at time of Object Creation itself

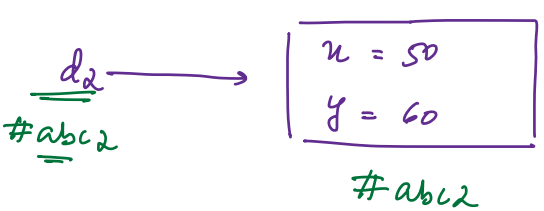
```
Class Node {
    int x, int y
    Node ( int p1, int p2 ) {
        x = p1 ; y = p2
    }
}
```

Constructor: Very similar to function
- No return type
- Constructor name same as class name

```
Node d = new Node (10, 20)
```




```
Node d2 = new Node (50, 60)
```



8:28 am → 8:37 am

Problems on linkedlist:

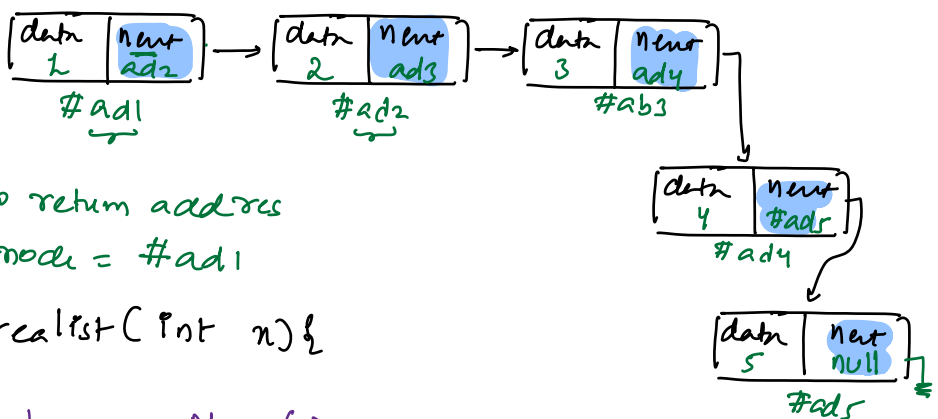
```
class Node{
    int data
    Node next;
    Node(int n){
        data = n;
        next = NULL
    }
}
```

```
Node n = new Node(10);
```

n → [abc#] → { data = 10, next = null } (abc#)

// Create linked list with n, nodes, each node data is from 1 - n, return head node of linked list?

Ex: n = 5

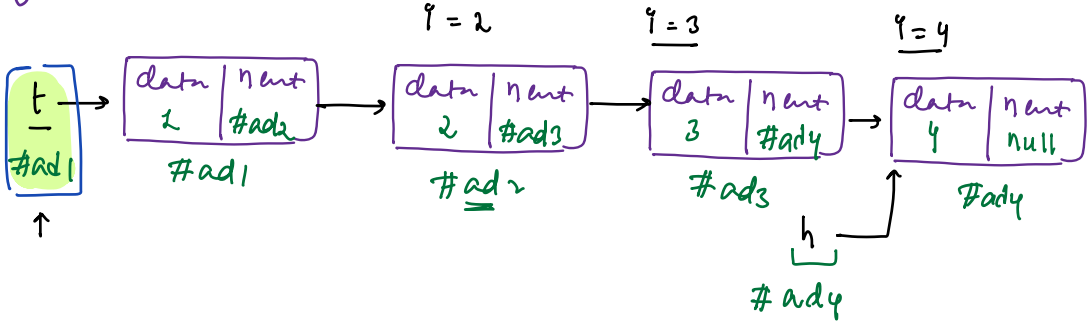


We need to return address of head node = #add1

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

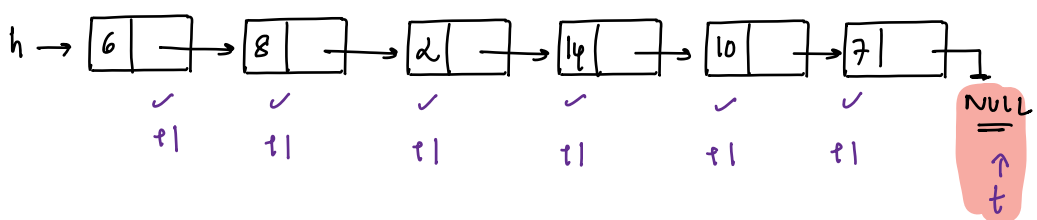
```
Node h = new Node(1)
Node t = h;
for(int i=2; i<=n; i++){
    h.next = new Node(i)
    h = h.next
}
```

Dry run



→

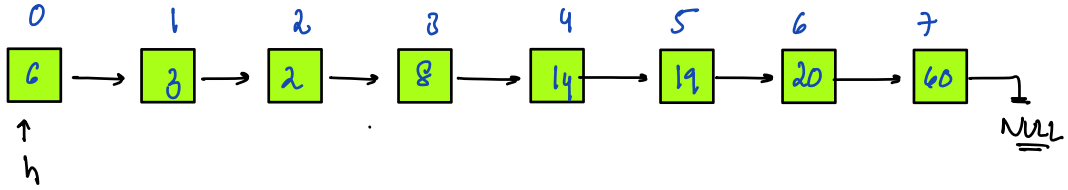
Q) Given head Node of linkedlist, return size



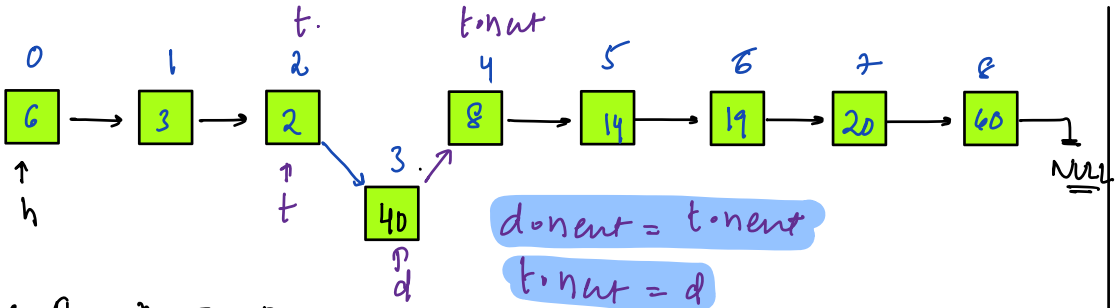

```
int size(Node h){
    Node t = h
    int c=0
    while(t != NULL){
        c=c+1
        t=t.next
    }
    return c;
}
```

TC: O(N) → indicate no: of nodes

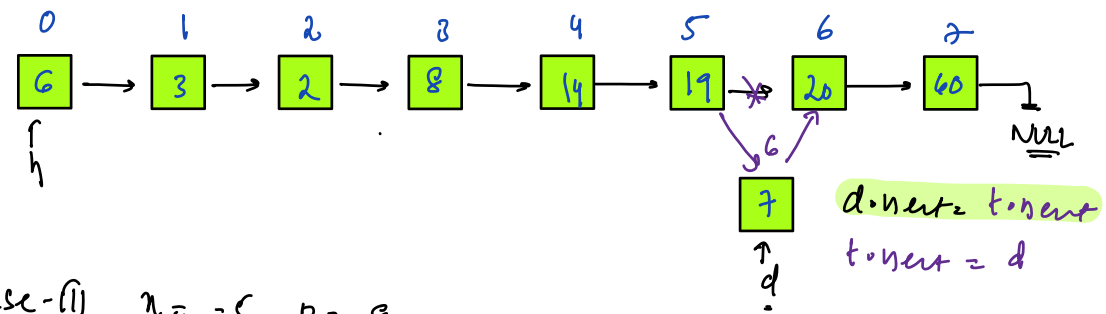
20) Given a linked list, insert a new node with data n at position p



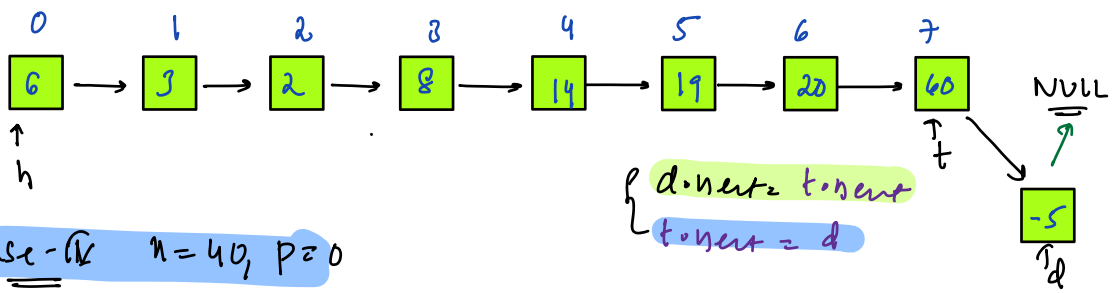
Case-I: n = 40, p = 3



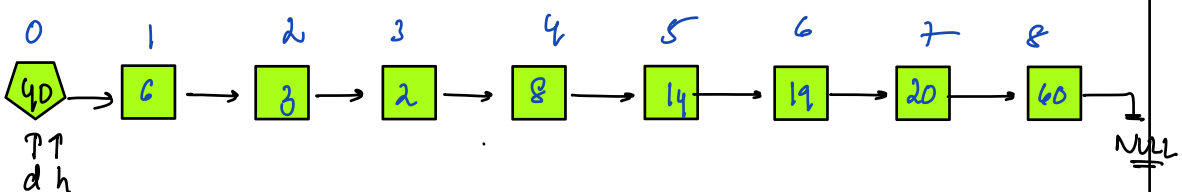
Case-II: n = 7, p = 6



Case-III: n = -5, p = 8

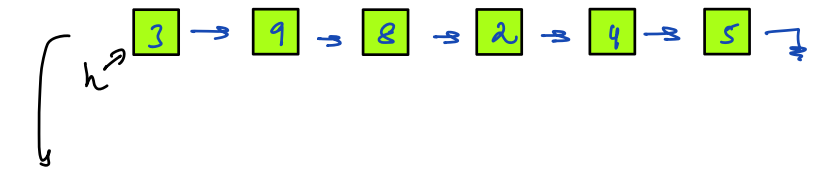


Case-IV: n = 40, p = 0



```
// head node change in linkedlist, return new head node
Node insertNode(Node h, int x, int p){

    Node d = new Node(x);

    if(p==0){
        d.next = h, h = d, return h;
    }

    Node t = h;
    // Take temp node to pos p-1
    // How many time we need to update t = p-1 time
    for(int i=1; i<p; i++){ => { i=[1, p-1] // p-1 times]
        t = t.next
    }

    d.next = t.next
    t.next = d

    return h;
}
```

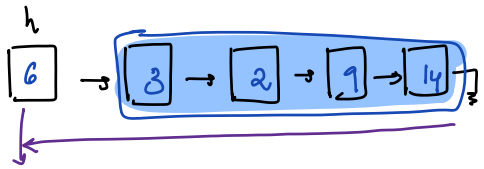
→ given linked list print reverse



Output: 5 4 2 8 9 3

Ass: Given head node print, entire linked list in reverse order

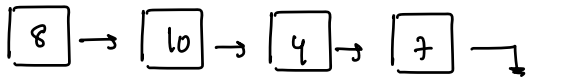
```
void printRev(Node h){
    if(h==null){return;}
    rev(h.next)
    print(h.data)
}
```



rev(h.next) = 14 9 2 3 6

print(h.data)

Trace:




```
void rev(Node h -> 8)
    if(h==null){return}
    rev(h.next)
    print(h.data)
```

TC: O(N)

SC: O(N)
↳ recursive stack size

```
void rev(Node h -> 10){
    if(h==null){return}
    rev(h.next)
    print(h.data)
```

```
void rev(Node h -> 4){
    if(h==null){return}
    rev(h.next)
    print(h.data)
```

```
void rev(Node h -> 7)
    if(h==null){return}
    rev(h.next)
    print(h.data)
```

output: 7, 4 10 8

```
void rev(Node h == NULL)
    if(h==null){return}
    rev(h.next)
    print(h.data)
```

Basics of Stacks / → Tuesday

Basic of Trees → Thursday