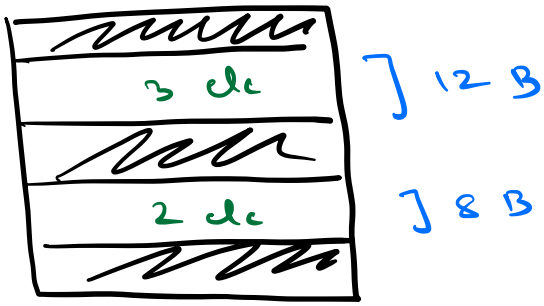
Issues with Array

int → 4 B

int a[5]

↓

20 Bytes



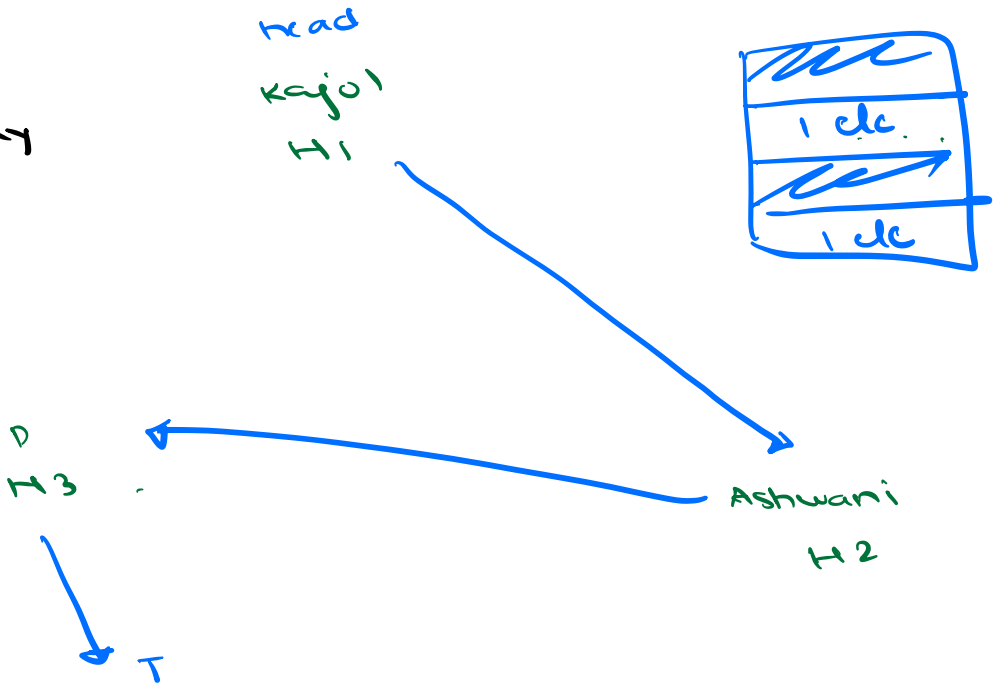
Linked List

↓

- linear data structure which can utilize free memory
- Do not need continuous space to store data of the linked list

① Array

② LL




```
class Node {
    int data
    Node next
}
```

Heap

x

10 K

10 K

data = 0

```
Node x = new Node( )
```

ref variable of node type

temp

10 K

```
Node temp = x
```

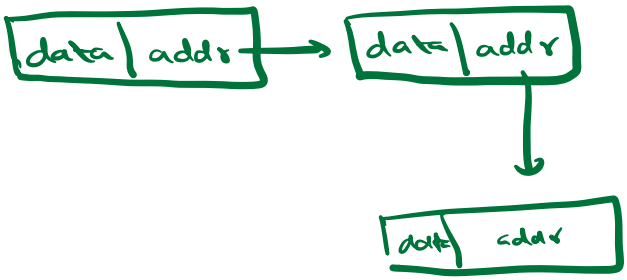
```
Node temp2
```

temp2

NULL

```
class Node {
    int data
    Node next

    Node (x) {
        data = x
        next = NULL
    }
}
```

Node




```
Node t = new Node (10)
```

t.data → 10

t.next → NULL

t

10 K

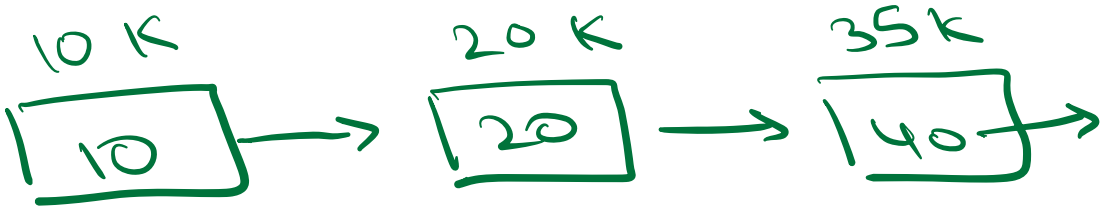

```
t.next = new Node (20)
```

35 K

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

10K

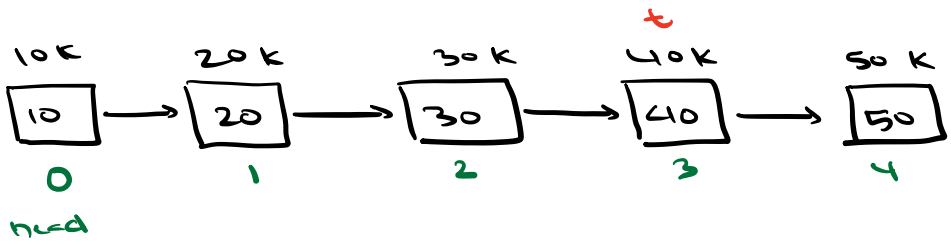
20K



head of LL (first node of LL)

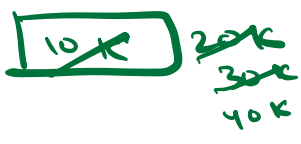
Operations on LL

K = 0
ans → 10



K = 3
ans → 40

temp



K = 3

i = 0 ✓ 20 K

i = 1 ✓ 30 K

i = 2 ✓ 40 K

1) Access k^{th} idx node

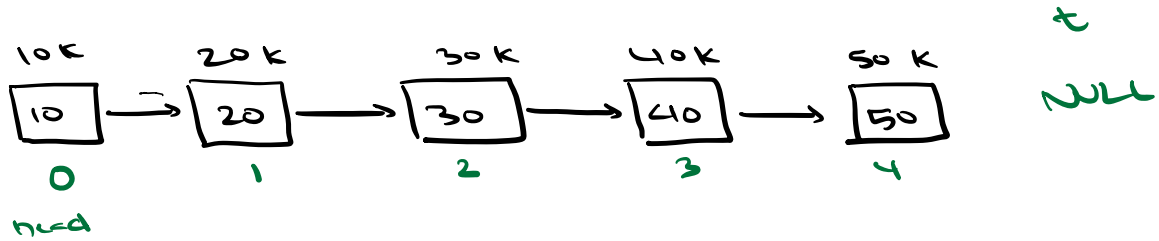
```
int Kth (Node head, int k) {
    Node temp = head

    for (i=0 ; i < k ; i++) {
        temp = temp.next
    }

    return temp.data
}
```

10 K

TC : O(K) → O(N)

SC : O(1)

2) Search for value x



x = 30 True

x = 15 False

```
                                10K      30
bool check (Node head, int x) {

        Node temp = head
        while (temp != NULL) {
            if (temp.data == x)
                return true
            else
                temp = temp.next
        }
        return false
}
```

TC : O(N)

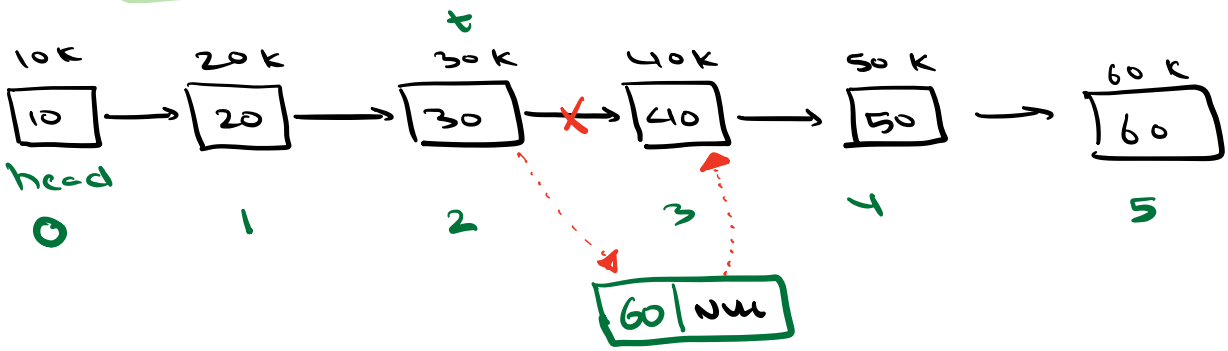
SC : O(1)

t → NULL

t.data → NPE

Null Pointer Exception

Q. Insert a new node with data v at index p



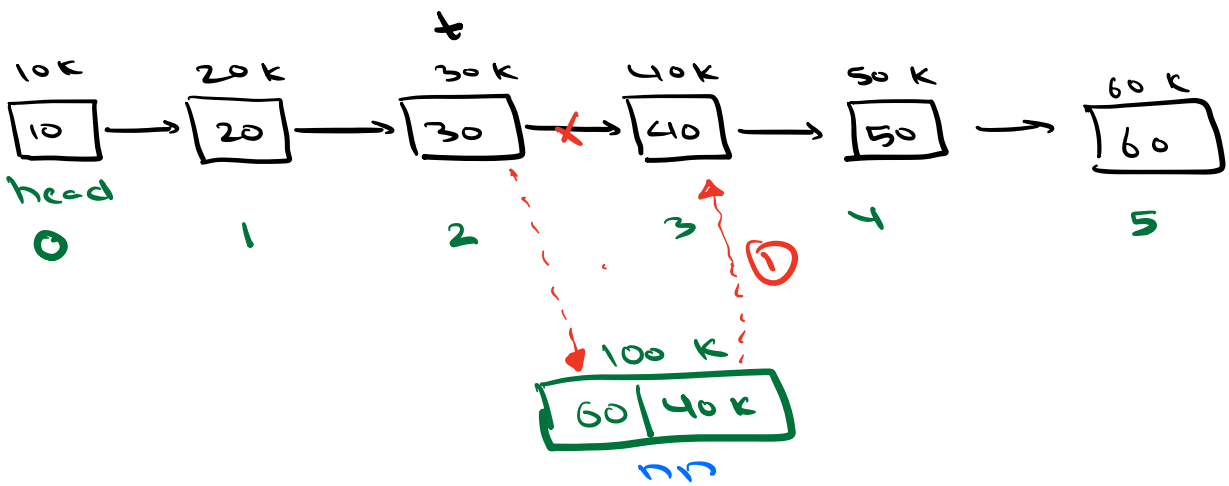
v = 60 p = 3

1. Create new node

Node nn = new Node(v)

2. Stop at $(p-1)^{th}$ node

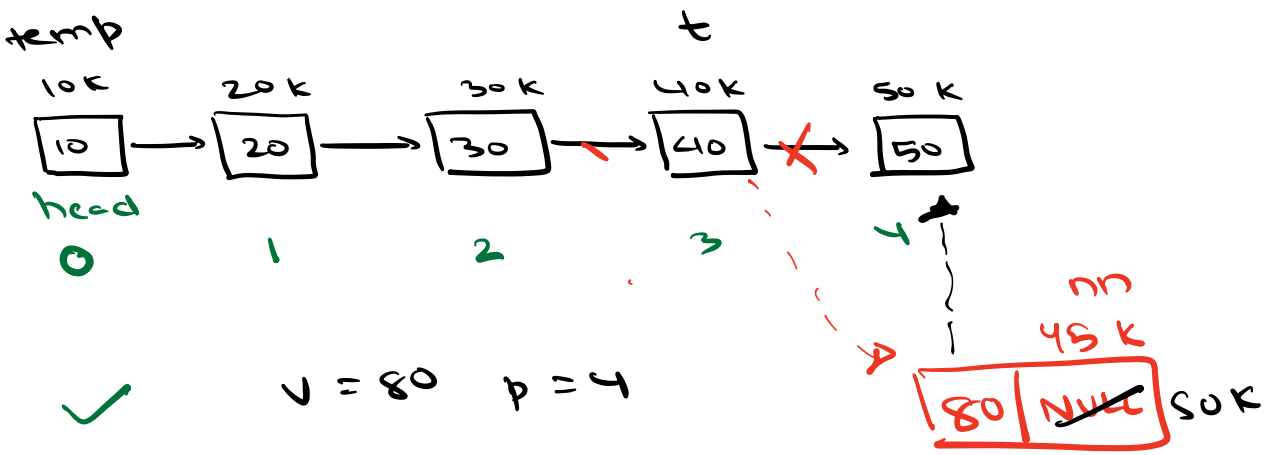
```
Node temp = head
for (i=0 ; i<p-1 ; i++) {
    temp = temp.next
}
```



```
Node nn = new Node(v)
Node temp = head
for (i=0 ; i<p-1 ; i++) {
    temp = temp.next
}
```

① nn.next = temp.next

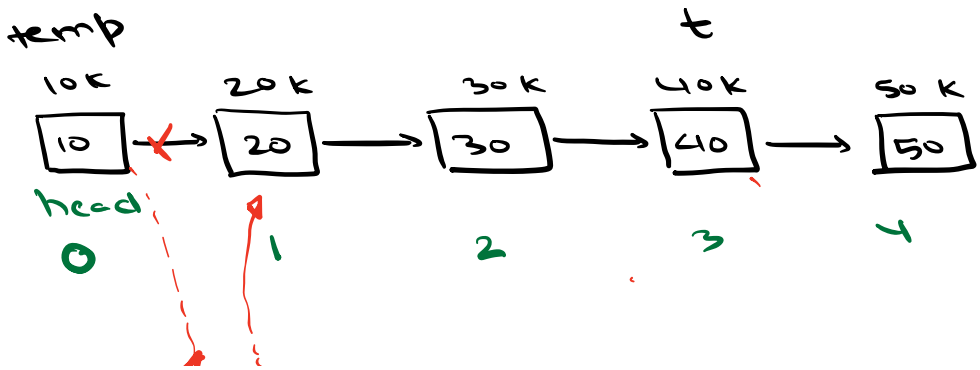
② temp.next = nn

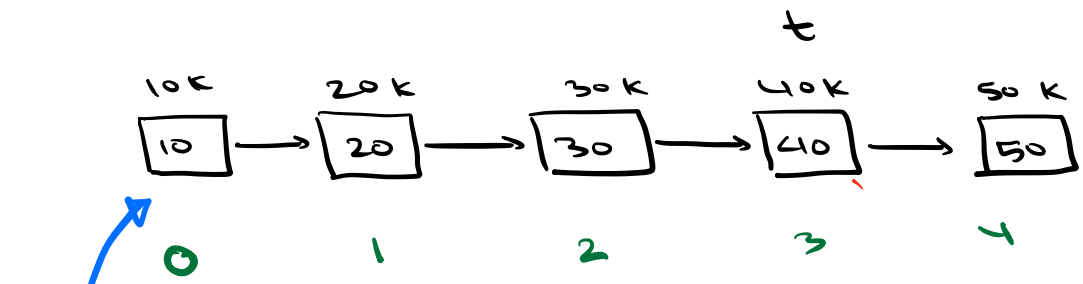


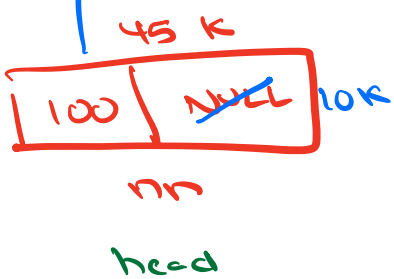
✓ v = 80 p = 4

Edge case

v = 100 p = 0







Node nn = new Node (v)

① nn.next = head

② head = nn

```
Node insertatk (Node head, int v, int p) {
    Node nn = new Node(v)
    if (p==0) {
        nn.next = head
        head = nn
        return head
    }
    Node temp = head
    for (i=0 ; i<p-1 ; i++) {
        temp = temp.next
    }
    ① nn.next = temp.next
    ② temp.next = nn
    return head
}
```

TC : O(p)

SC : O(1)

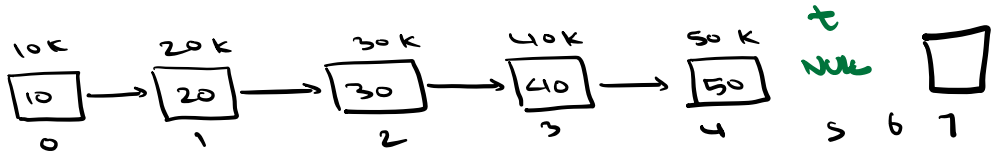
// Dry run p = 80 v = 50

insertion at last

---

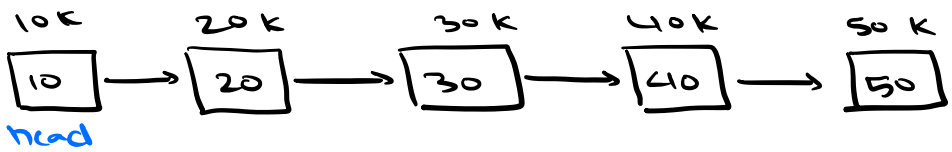
v = 10 p = 8 jump → p-1 times

11:05

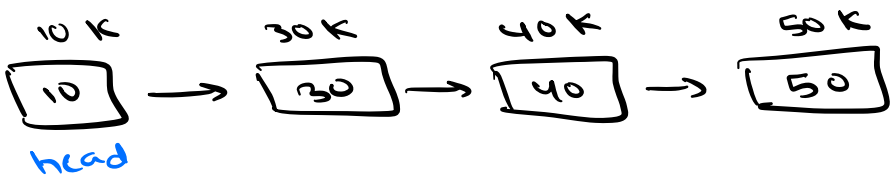


p → 0, 1, 2, 3, 4, 5 last

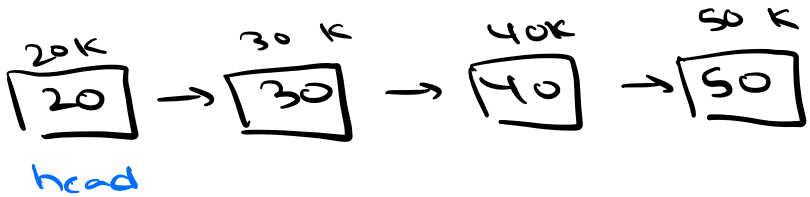
Deletion in a LL given X

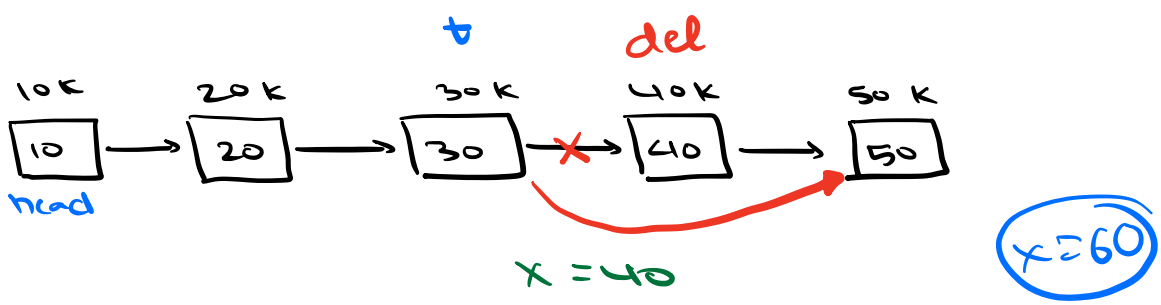


eg 1 $x = 20$



eg 2 $x = 10$

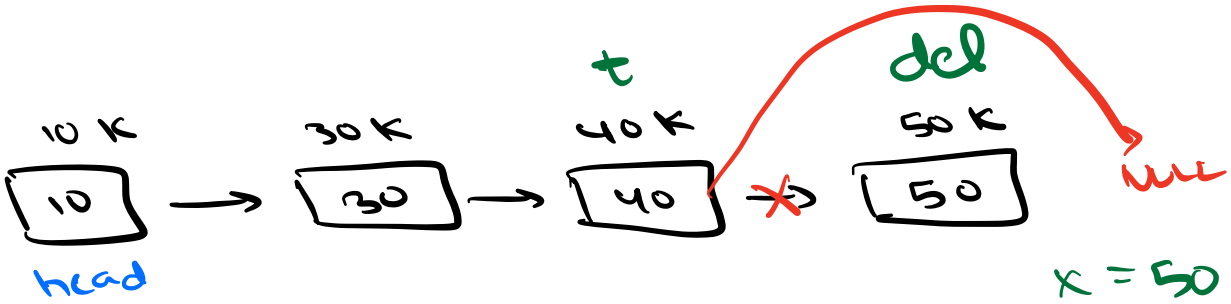





```
Node temp = head
while (temp.next != NULL) {
    if (temp.next.data == x) {
        Node del = temp.next
        temp.next = del.next
        free(del)
    }
```

t.next's data = x

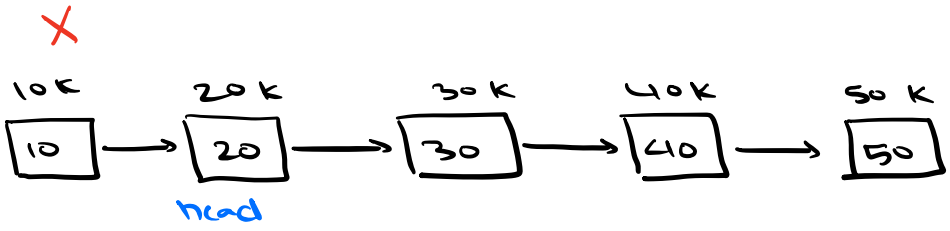
temp = temp.next




```
Node temp = head
while (temp.next != NULL) {
    if (temp.next.data == x) {
        Node del = temp.next
        temp.next = temp.next.next
        free(del)
        return head
    }
    temp = temp.next
}
```

t.next's data = x

C++



x = 10

```
if (head.data == x) {
    Node del = head
    head = head.next
    free(del)
```

```
}    return head
```

```
Node delete(Node head, int x) {
    if (head == NULL) return NULL
    if (head.data == x) {
        Node del = head
        head = head.next
        free(del)
        return head
    }

    Node temp = head
    while (temp.next != NULL) {
        if (temp.next.data == x) {
            Node del = temp.next
            temp.next = temp.next.next
            free(del)
            return head
        }
        temp = temp.next
    }
    return head
}
```

t.next's data = x

C++

→ no deletion happened

TC: O(N)

If LL is empty, head = NULL

Edge cases : LL is empty | → head = NULL
size is 0

size is 1

head.next = NULL

Operation should work
on 0th / last node

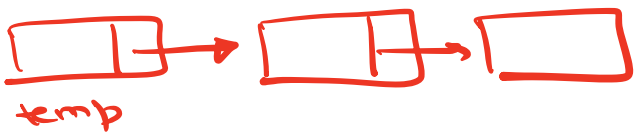
Not null

___ . data
. next

temp . next . next . next
≠ NULL

≠ NULL

≠ NULL



$\boxed{\text{Data} \mid \text{addr}}$

int → 4B

4 B → 32 bit processor

8 B → 64 bit processor