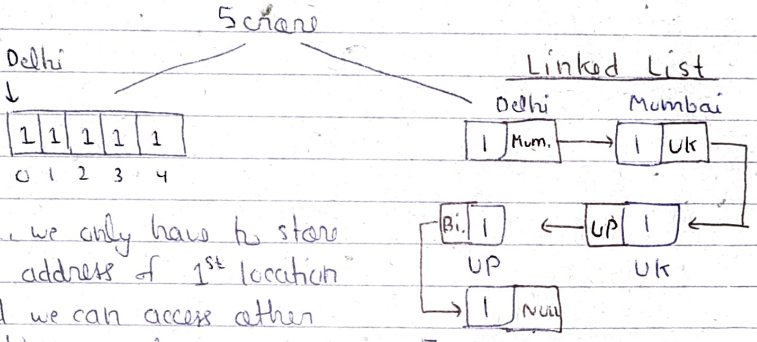
Date: / /
Page:

Day - 112

Linked List

* **Linked List:**

⇉ To understand LL, suppose we have 5 crore and we have to hide it and the constraint is that, we have to buy 5 houses —



⇉ Here, we only have to store the address of 1st location and we can access other locations easily.

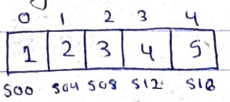
⇉ But here that's not possible

⇉ So, in this way, we only have to store the address of Delhi & we can easily ~~address~~ go to other locations.

⇉ So, this is Linked List.

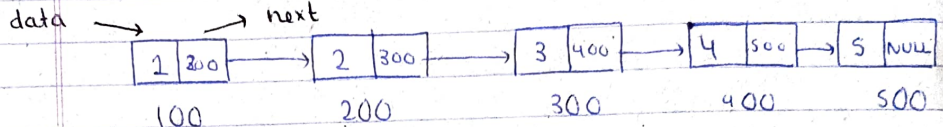
⇉ Now, suppose we want to store 1, 2, 3, 4 & 5.

⇉ Then in array :



Date: / /
Page:

⇒ And in Linked List,



⇒ This is singly Linked List.

Code

```
class Node {
    public:
    int data;
    Node * next;
};

int main(){
    // Node A1;
    // A1.data = 4;
    // A1. data = NULL;
    // we have create Node in dynamic manner
       because we don't know how mch node will be
       required.
    Node * Head;
    Head = new Node();
    Head -> data = 4;
    Head -> next = NULL;
}
```

⇒ we can also create constructor.

```
class Node{
    public:
    int data;
    Node * next;

    Node(int value){
        data = value;
        next = null;
}

int main(){
    Node * Head;
    Head = new Node(4);
}
```

* Operations in Linked List:

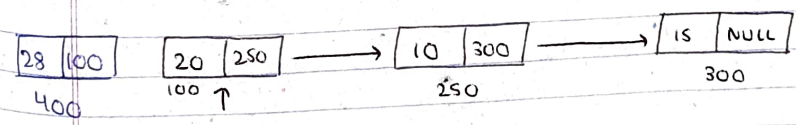
⇉ Insertion

⇉ Where we can insert any node —

→ Start.

→ End.

→ Middle



Head

⇉ Head is a pointer that points to the starting node of the LL.

⇉ It stores the address of first node.

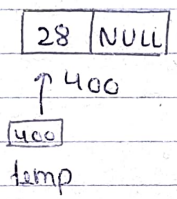
Date: / /
Page:

=> Now, we want to add node to the LL.

=> Firstly, we have to create that node.
So,

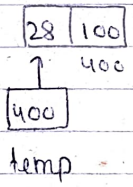
```
Node * temp;
temp = new Node(28);
```

=>

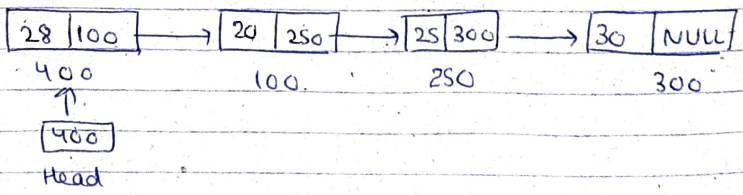


=> Now, if we want to connect it with LL then, first we have point its next to the Head of the LL.

```
temp -> next = Head;
Head = temp;
```



=> After that, Head will point this newly created node.

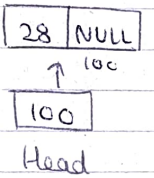


Date: / /
Page:

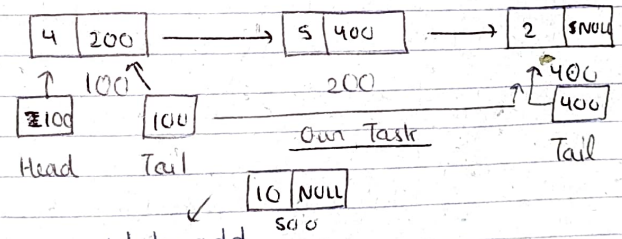
=> Suppose, we don't have any LL.

Head

```
Node *Head;  Head = NULL;
if (Head == NULL){
    Head = new Node(28);
}
```



=> Now, we want to add node at the end of the LL.



=> we want to add this node.

=> Note: We don't move 'Head' to perform our operations.

=> First we create a Tail pointer.

=> After that, we will move it to the end of LL.

```
while(Tail -> next != NULL){
    Tail = Tail -> next;
}
```

Date: / /
Page:

```
Node * temp;
temp = new ~~no~~ Node(10);
tail → next = temp;
```

⇒ When LL doesn't exist —

```
if (Head == NULL){
    Head = new Node(10);
}
```

⇒ Now we have a list → |2|4|6|8|, And we will add every element in the end of the L.list.

⇒

```
int main(){
    Node * Head = NULL;
    for( i=0; i<4; i++){
        if( Head == NULL)
            Head = new Node(arr[i]);
        else {
            Node * tail = Head;
            while (tail → next != NULL){
                tail = tail → next;
            }
            Node * temp = new node (arr[i]);
            tail
            ~~temp~~ → next = temp;
        }
    }
}
```

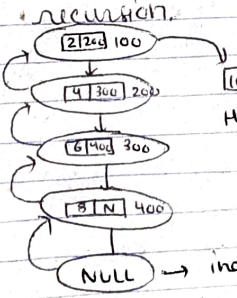
Date: / /
Page:

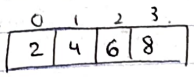
⇒ Here, we are again and again traversing whole LL to shift the tail at their correct position.

⇒ So, we can take an extra variable to solve this problem.

```
int main(){
    Node * Head = NULL;
    Node * Tail = NULL;
    for(i=0; i<4; i++){
        if(Head == NULL){
            Head = new Node(arr[i]);
            Tail = Head;
        }
        else{
            Tail → next = new Node(arr[i]);
            Tail → ~~next~~ = Tail → next;
        }
    }
}
```

⇒ Now our task is to add node at End by using recursion.



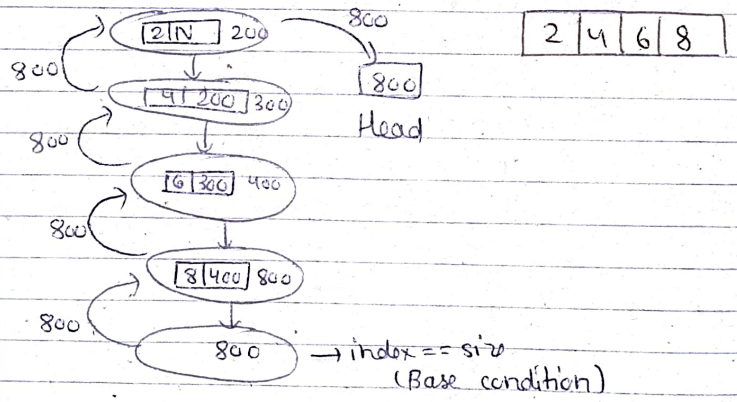


Date: / /
Page:

```
Node* createLL(int arr[], int index, int size){
    if(index == size)
        return NULL;
    Node *temp;
    temp = new Node(arr[index]);
    temp->next = createLL(arr, index+1, size);
    return temp;
}
```

⇒ Now, our task is to create LL by adding elements in the front by using recursion.

⇒

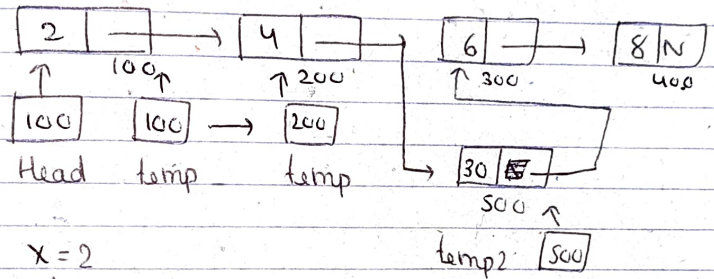


⇒

```
Node* createLL(int arr[], int index, int size, Node *prev){
    if(index == size)
        return prev;
    Node *temp;
    temp = new Node(arr[index]);
    temp->next = prev;
    return createLL(arr, index+1, size, temp);
}
```

Date: / /
Page:

⇒ Now, we have to insert the node at a given ~~index~~ position.



⇒ x = 2
value = 30

⇒
```
temp2 → next = temp → next;
temp → next = temp2;
```