☆ Doubly LinKed lists

-> How create Doubly LinKed lists :-

(i)
```
class Node {
    this.value = Value
    this.next = null
    this.prev = null
}
```

```
class DoublyLinkedLists {
    constructor(Value){
```

this.hea

```
Const = newNode = new Node (value)
  this.head = new Node
  this.tail = newNode
  this.length = 1
}
```

☆ Push method in (DLL):-

→ there is two case :-

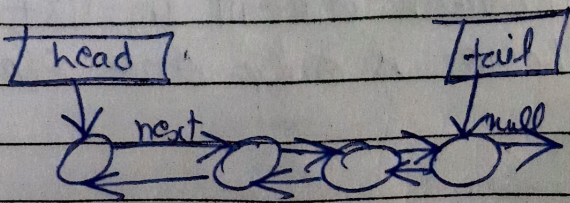
(i) if list is empty.

(ii) if there is element.

(i)
```
Const newNode = new Node (value)
if (!this.head){
  this.head = newNode;
  this.tail = newNode;
}else{
```

(ii)
```
  this.tail.next = newNode
  newnode.prv = this.tail;
  this.tail = newNode;
}
this.length ++;
return this;
```

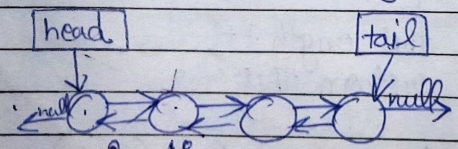
☆ Pop method in (DLL)



→ There is 3 case :-
(i) if List is empty
(ii) if there is only one element in List
(iii) More than one element

(i)
```
if (!this.head){
    return "List is empty" }
if (this.length===1){
    this.head = null
}else{
    let temp = this.tail
    this.tail = temp.prev
    this.tail.next = null
    temp.prev = null
}
this.length--;
return temp;
```

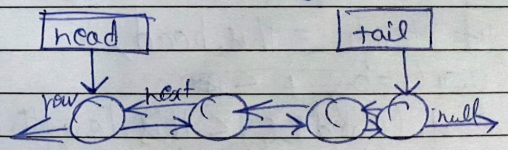
☆ Deletion at beggininming in (DLL):-



(i) if there is List is empty
(ii) if there element in list
(iii) if there is only one element

```
if (!this.head) return "List is empty"
let temp = this.head;
if (this.length===1)
    this.head = null;
    this.tail = null;
}else{
    this.head = this.head.next
    this.head.prev = null
    temp.next = null
}
this.length--;
return temp;
```

☆ Insertion at Begginning

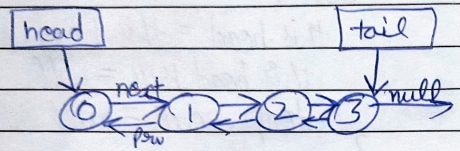


(i) if there is no element in List.
(ii) if there is more than one element in List.

(i)
```
const newNode = new Node(value);
if (!this.head){
    this.head = newNode;
    this.tail = newNode;
}else{
    newNode.next = this.head;
    this.head.prev = newNode;
    this.head = newNode;
}
```

```
this.length++;
return this;
```

☆ Search an Node in Doubly LinKed List through index



(i) if index out of bound

```
if (index <0 || index >= this.length){
    return 'Index out of Bound'
}
```

(ii)
```
let temp = this.head;
for(i=0; i<this
  if (index < this.length/2){
    for(i=0; i<index; i++){
      temp = temp.next;
    } else {
    temp = this.tail;
    for (let i = this.length-1; i>index; i--){
      temp = temp.prev;
    }
  return temp;
}
```

☆ Insert Node at index in (DDL) :-

⇒ There are four cases :-

(i) `if (index ===<0) return this.insertAtBeggin(val);`

(ii) `if (index === this.length) return this.push(val)`

(iii) `if (index<0 || index > this.length) return "Index out of Bound"`

(iv)
```
const newNode = new Node(value)
const befor = this.searchANode(index-1)
const after = before.next;
  before.next = newNode;
  newNode.prev = before
  newNode.next = after
  after.prev = newNode;
  this.length++;
  return true;
```

☆ RemoveNodeAtIndex in Doubly Linked List

(i) if index is 0.

```
if (index === 0) return this.deletionAtBeg();
```

(ii) if index is at total length of (DLL)

```
if (index === this.length) return this.pop();
```

(iii) if index is Index out of Bound

```
if(index<0 || index>=this.length){
  return 'Index out of Bound'
const temp = this.searchAnNode(index);
```

```
temp.prev.next = temp.next;
temp.next.prev = temp.prev;
temp.next = null;
temp.prev = null;
this.length--;
return temp;
}
```