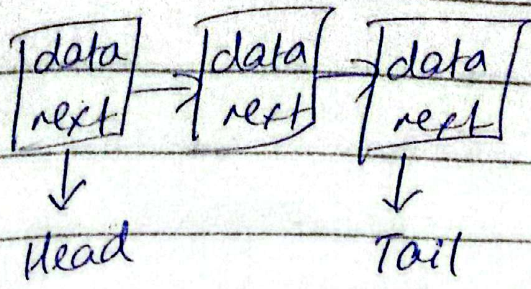
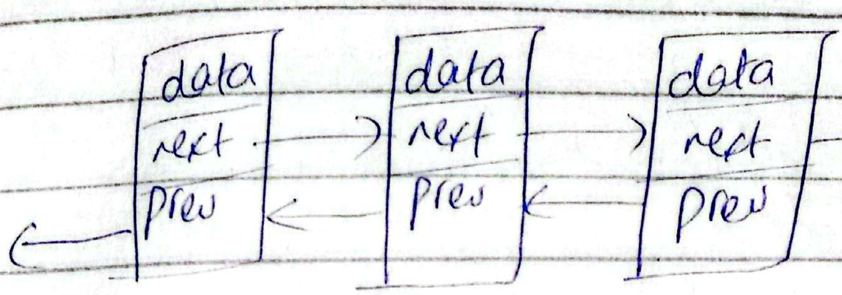
# Doubly Linked List

① • Singly LL
② • Doubly LL
③ • Circular LL



→ Singly Linked List



→ Doubly LL

→ Node Formation

```
class Node{
  int data;
  Node* next;
  Node* prev;
}
```

Nodes/objects handled by Doubly lists

```
class DoublyList{
  Node* head
  Node* tail
}
```

Functions

→ push_back/front (Adds on the front)

① Create new Node

LL.pushfront

Empty

```
if(head==NULL){
head=tail=newNode
```

Non-Empty

```
else
newNode->next=head
head->prev=newNode
head=newNode
```

→ Printing

```
void Print(){
Node* temp = head;
while(temp!=NULL){
 cout<<temp->data;
  temp = temp->next;
 }
```

→ Push back (Adding on the back)

① Creating newNode

```
        Node* newNode = newNode(val)
        Push back(2)
```

Empty

```
if(head==NULL){
head=tail=NewNode
 }
```

Non Empty

```
else{
NewNode->prev=tail
tail->next=newNode
 tail = NewNode
```

→ Pop front (Removing the element from the front)

```
    Node* temp= head
    head=head->next
    if(head!=NULL){
     head->prev=NULL;
    temp->next = NULL
     delete temp;
```

Base case

```
if(head==NULL)
"Linked list is
 Empty"
```

pop_back( Removes element from the back)

```
Node * temp = tail
if( tail != NULL){
  tail->next = NULL
  temp->prev = NULL
    delete temp;
```

if(head == tail == NULL)

" Empty linked list"