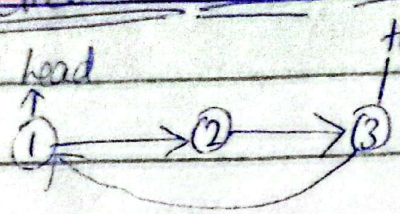
→ Circular Linked List



Head = tail->next

Node Formation

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

Nodes/objects in Circular List

```
class Circularlist{
Node* head;
Node* tail;
}
```

Function

→ Insert at Head (Adding at the head of list)

① Create the Node

```
Node* newNode = newNode
```

② IAH(1)

head==NULL

```
if(head==NULL){
head=tail=newNode
tail->next=head
```

head nonempty

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

↓ without the use of Head

```
newNode->next = tail->next
tail->next = newNode
```

→ Printing

```
if(head==NULL){
return;
}
cout<<head->next<<"->"
Node* temp = head->next
```

```
while(temp!=head){
cout<<temp->data<<"->";
temp = temp->next
}
cout<<temp->data;
```

→ Insert at tail

① Creating the Node (Node* newNode = new Node(val))

② IAT(1)

Empty

```
if(head==NULL){
head=tail=newNode
tail->next=head
}
```

NonEmpty

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

→ Delete at Head

① case — Empty

```
if(head==NULL){
return
```

② case — Single Node

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

③ case — Multiple Nodes

```
Node *temp=head
head=head->next
tail->next=head
temp->next=NULL
delete temp
```

→ Delete at tail

① case — Empty

```
if(head==NULL)
{
return
}
```

② case — Single Node

```
else if(head==tail){
delete head
head=tail=NULL
}
```

③ case — Multiple Nodes

```
else
{
Node* temp=tail
Node* prev=head
while(prev->next!=tail){
prev=prev->next
tail=prev
tail->next=head
temp->next=NULL
delete temp
```