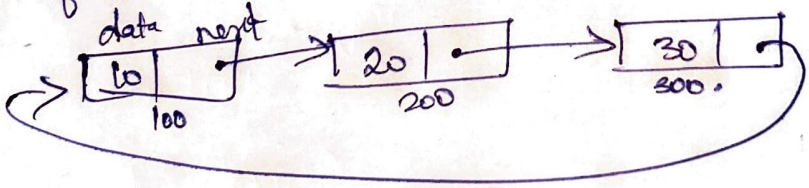
Circular Linked List:-

There are 2 types of circular linked list
- Singly circular linked list
- Doubly circular linked list.

* Circular linked list - It is a Data structure where the last node connects back to the first forming a loop.
* Continuous traversal without any interventions.
* Scheduling, Managing playlist, Task maintenance
* 2 types
  - Circular singly linked list
  - Circular doubly linked list.
* Traverse - until we reach where we started.

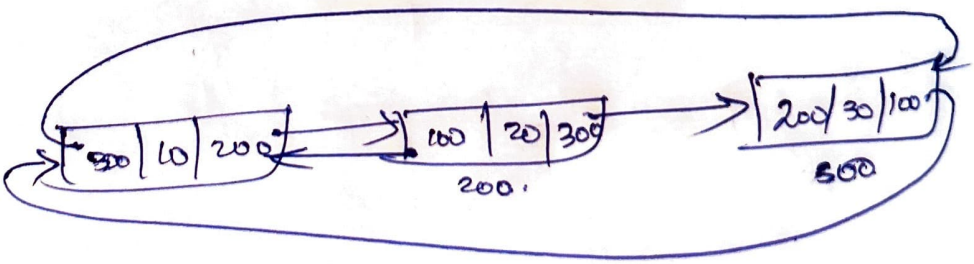
Circular Singly linked list:-

The pointer of the last node holds the address of first node.



Circular Doubly Linked List:-

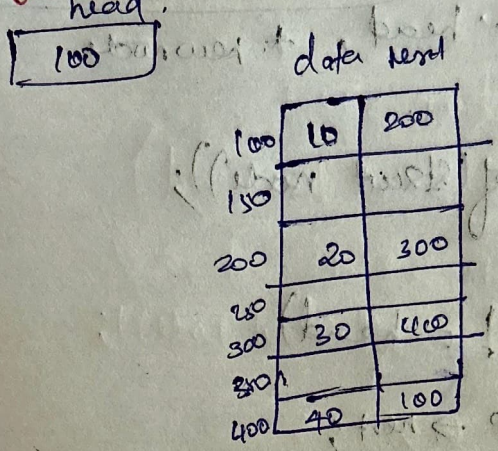
Last node stores the address of first node
first node stores the address of last node.



Application of Circular Linked List.

* Browser surfing :- Record of pages visited by user in past are saved.
(prev ← button, on clicking it the past history web pages are shown)

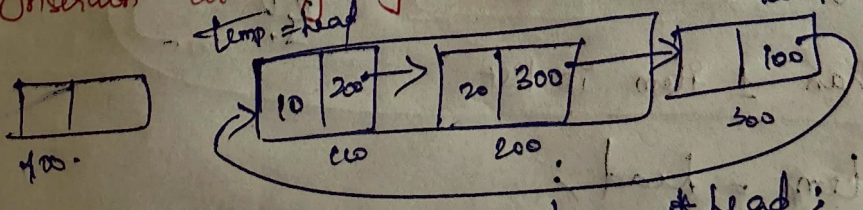
Memory Representation:



Operations in Circular singly linked list

* Insertion at begning
* Insertion at end
* Deletion at begning
* Deletion at end
* Searching
* Traversal

Insertion at beginning.

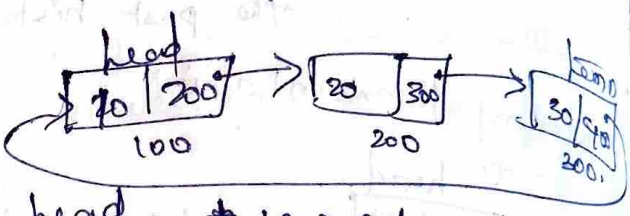



```
struct node *newnode, *temp, *head;
temp = head;
while (temp -> next != head)
    temp = temp -> next;
newnode = malloc(sizeof(struct node));
newnode -> data = 10;
```

```
temp -> next = newnode;
newnode -> next = head;
head = newnode;
```

```
if (head == NULL)
{ head = newnode;
  newnode -> next = head;
}
```

Insertion at end:




```
struct node *temp, *head, *newnode;
temp = head;
newnode = malloc(sizeof(struct node));
newnode -> data = 40;
while (temp -> next != head)
    temp = temp -> next;
newnode -> next = temp -> next;
temp -> next = newnode;
newnode -> next = head;
```

Deletion at begning:-

(No item in list)

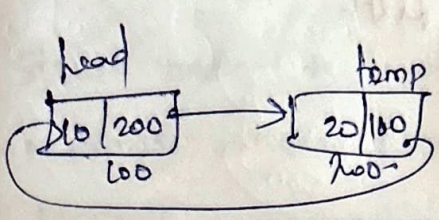
```
if (head == NULL)
    printf("underflow");
```

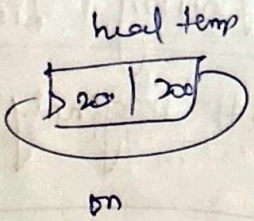
(only one item in list)

```
if (head -> next == head)
{ head = NULL;
  free(head);
}
```

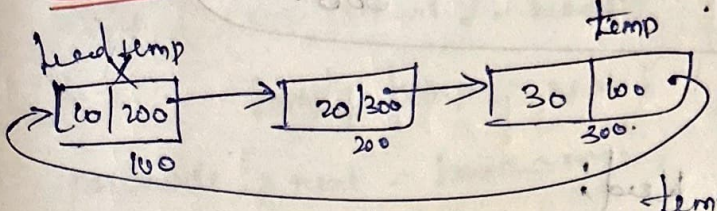
More than 1 item in list

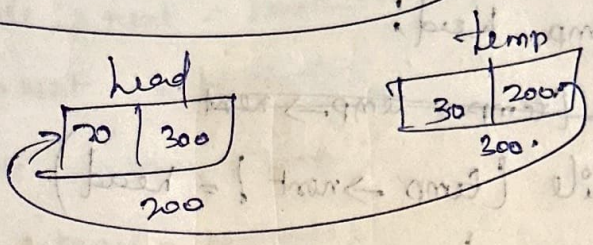
```
temp = head;
while (temp -> next != head)
    temp = temp -> next;
temp -> next = head -> next;
free(head);
head = temp -> next;
```



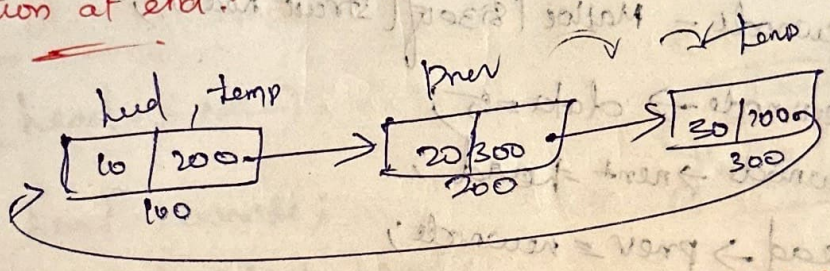


More than 1 item in list:





Deletion at end:



List is empty

```
if (head == NULL)
printf("\n underflow");
```

List contain only one element

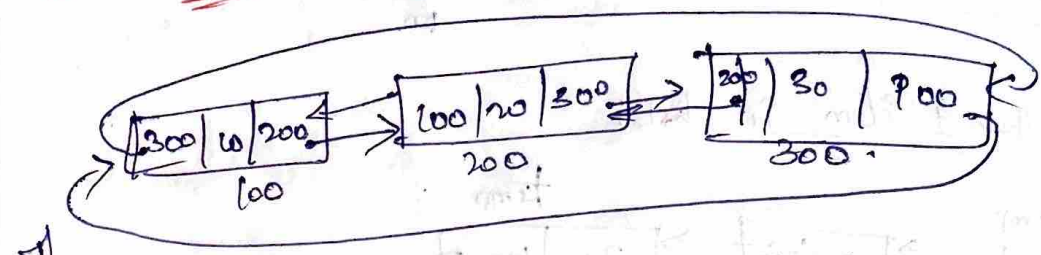
```
if (head -> next = head)
    head = NULL;
    free(head);
}.
```

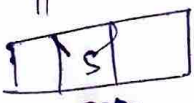
List contains more than one element:

```
temp = head;
while (temp -> next != head)
{
    prev = temp;
    temp = temp -> next;
}
Prev -> next = head;
free (temp);
```

Circular Doubly Linked

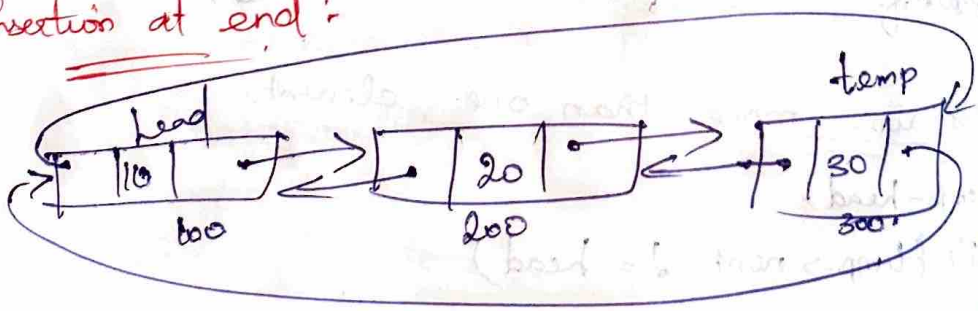
Insertion at beginning:






```
temp = head;
while (temp -> next != head)
    temp = temp -> next
newnode = Malloc (sizeof (struct node));
newnode -> data = 5;
newnode -> next = head;
head -> prev = newnode;
temp -> next = newnode;
newnode -> prev = temp;
head = newnode;
```

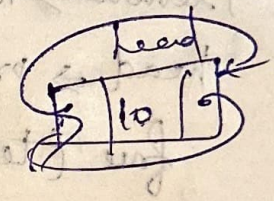
Insertion at end:



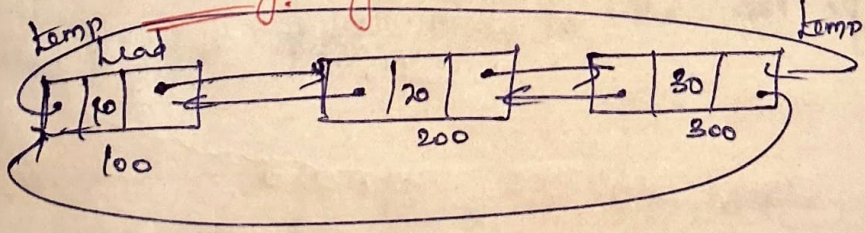
```
Struct node * head, *temp, * newnode;
newnode = Malloc (sizeof (struct node));
newnode -> data = 50;
temp = head;
while ( temp -> next != head)
        temp = temp -> next;
newnode -> next = temp -> next;
temp -> next = newnode;
newnode -> prev = temp;
head -> prev = newnode;
```

When list is empty

```
if ( head == NULL)
{
  head = newnode;
  newnode -> next = head;
  newnode -> prev = head;
}
```

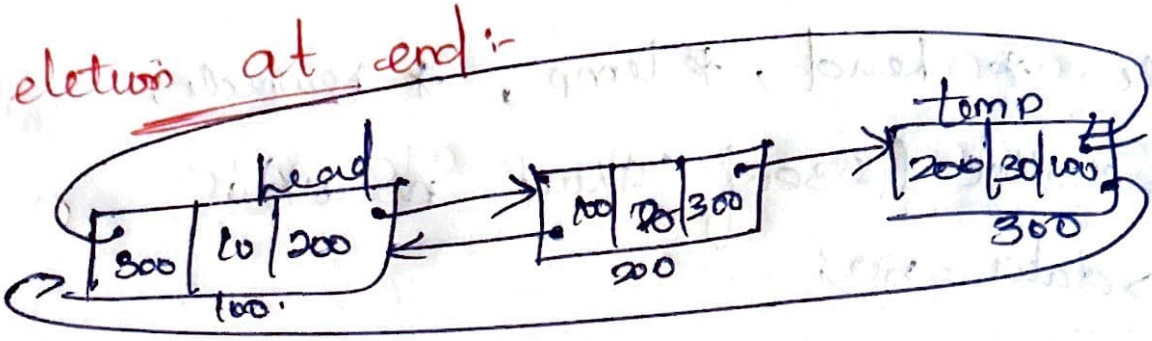


Deletion at beginning:-




```
temp -> next = head -> next;
head -> next -> prev = temp;
free (head);
head = temp -> next;
```

Deletion at end :-




```
struct node *head, *temp, *previous;
temp = head;
while (temp -> next != head)
  {
     prev = temp;
     temp = temp -> next;
  }
// * temp -> prev -> next = head; *//.
previous -> next = head;
head -> prev = previous;
free (temp);
```