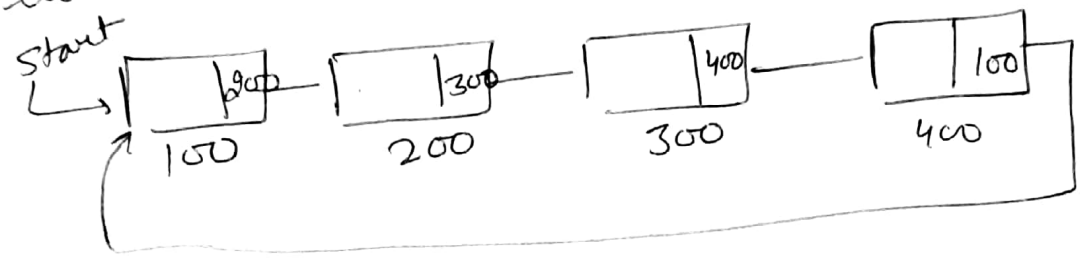
# Lecture - 19

## Circular linked list

A link list whose last node points back to the first node instead of containing the null pointer is called a circular linked list.

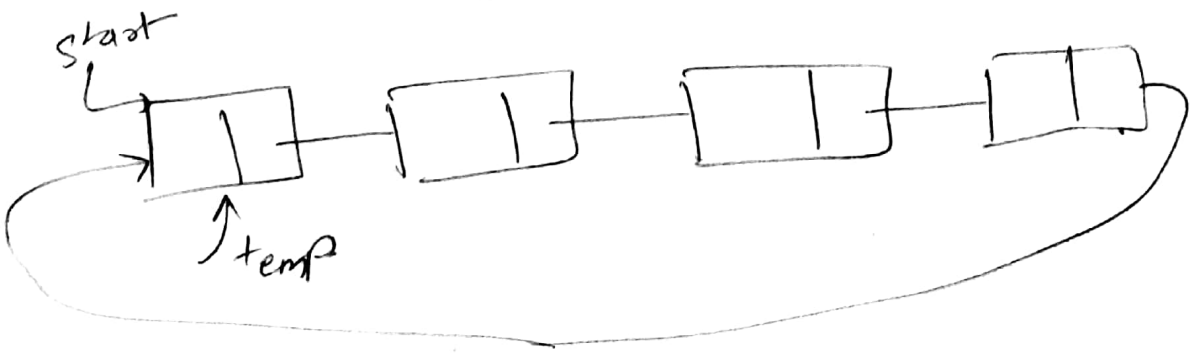



## Creation of circular list :-

- Follow the same step of creation as we did in single link list creation.
- Just assign the address of first node to the link of last node
  i.e say when we create the last node by a pointer say temp
  then after that make an additional stmt as

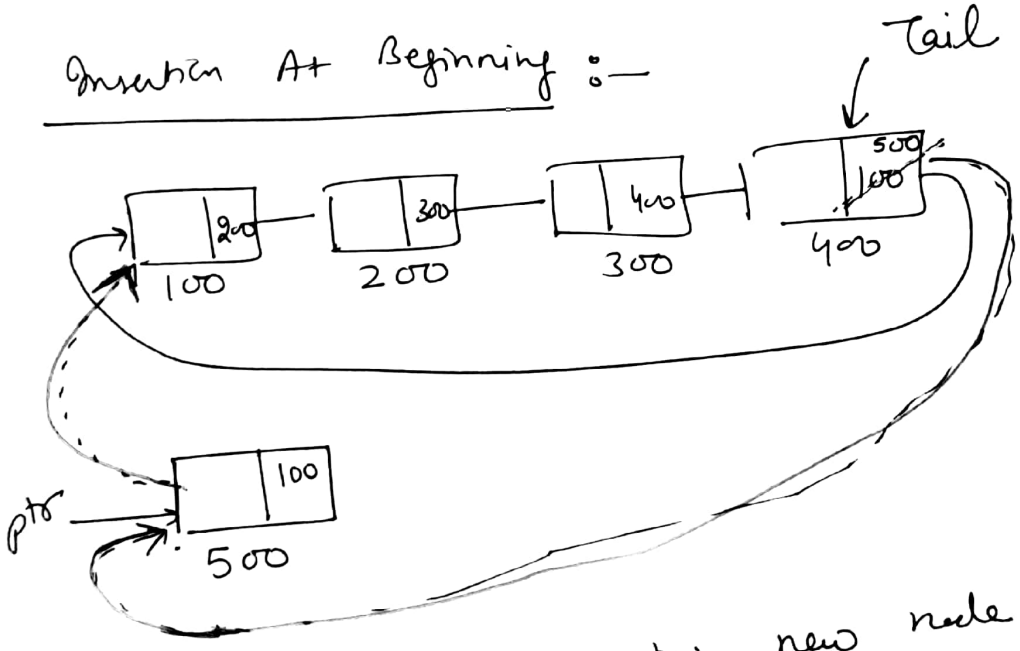
```
temp -> next = start;
```

## For Traversing of list :-



```
display ()
{
node * temp = start;
while ( temp -> next != start) || Here if we write temp != start
{                                 Then this condition will be false for the first time itself.
printf ("%d", temp -> data);
temp = temp -> next;
}
printf ("%d", temp -> data); || to print the data of last node
```

Insertion At Beginning :—

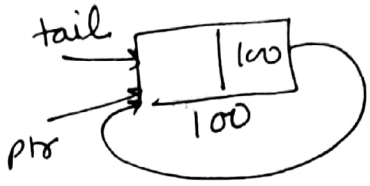



We want to insert this new node at the Beginning. Instead of start pointer we are considering a Tail pointer pointing at last node is holding the address of last node

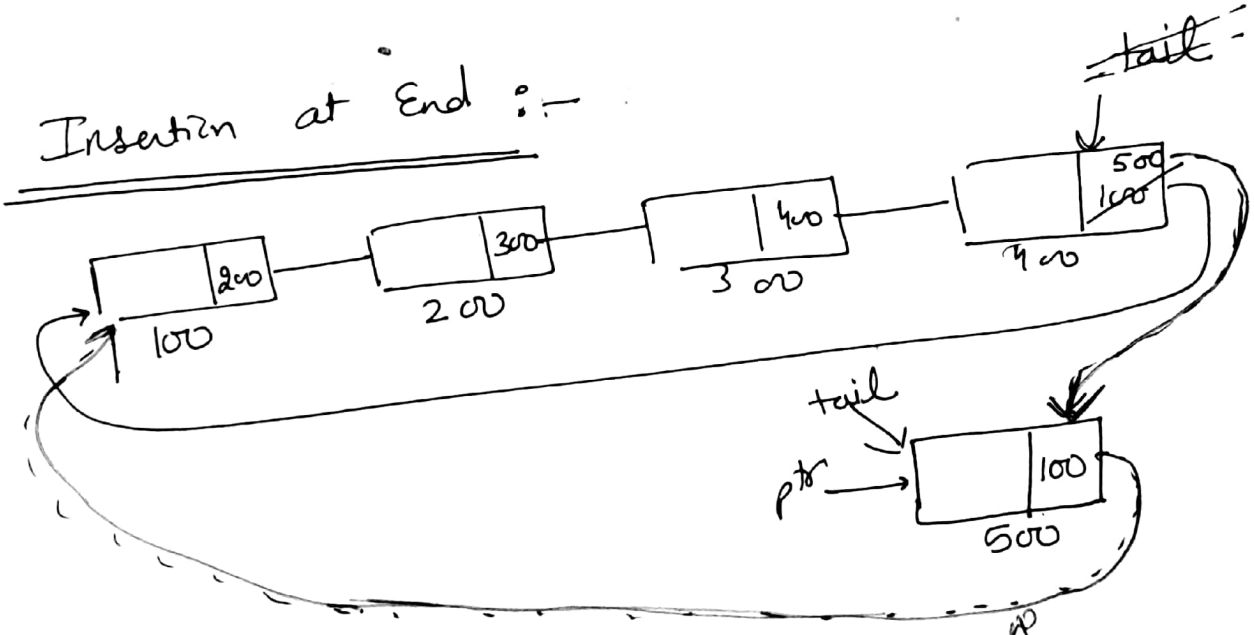
I am just showing the logic part

```
if (tail == 0) // if (tail == NULL)
{
    tail = ptr;
    tail -> next = ptr;
}
else
{
    ptr -> next = tail -> next
    tail -> next = ptr
}
```

// it becomes the first node




## Insertion at End :-




```
if ( tail == 0)  // special case
{
    tail = ptr;
    tail -> next = ptr;
}
```

Blue line show the updated list

Else

{

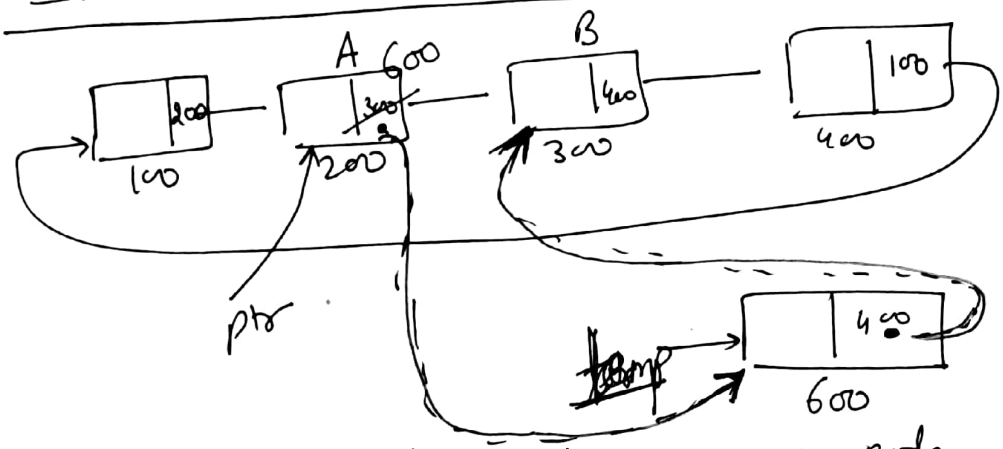
ptr → next = tail → next;

tail → next = ptr;

tail = ptr;

}

Insert at In-between :-




Say we want to insert one new node between A & B.

~~This can be done by Comparing the~~ Or we can find the location by asking the position no. from user.

Say we ask the user at which position you want to insert.

- Calculate the length of the list

```
l = getlength();
read the pos.
if ( pos < 0 || pos > l)
{ invalid position
}
```

```
else if (pos == 1)
    {
    insert at Beginning
    }
temp = ( ) malloc ( ) ;  || create a new node
ptr = tail → next ;  || initialize a pointer to traverse the list. Now ptr is at first node

while ( i < pos-1)
 {
  ptr = ptr → next;  || update ptr
   i++;
 }
temp → next = ptr → next;
   ptr → next = temp;
 }.
```