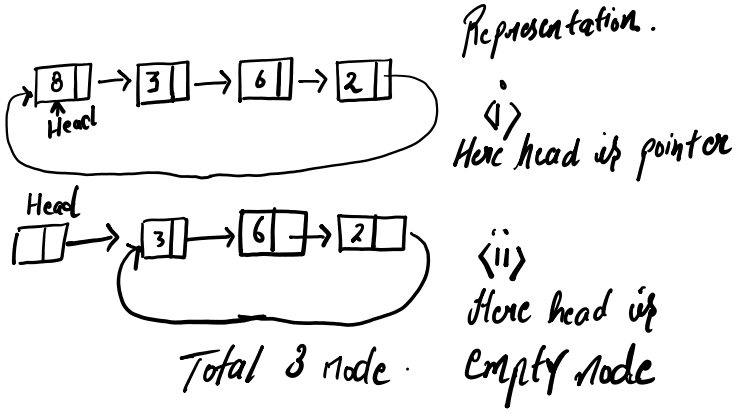
Circular & doubly LL

29 June 2022 15:06

Circular linked list (CLL)



Representation.

(i) Here head is pointer

(ii) Here head is empty node

Total 3 node.

Display CLL

☆ Function of Display

```
Void display (node* p)
{
    node *temp;
    temp = p;
        do {
        cout << p -> data << " ";
        p = p->next;
    } while (p != temp)
}
```

Using recursion

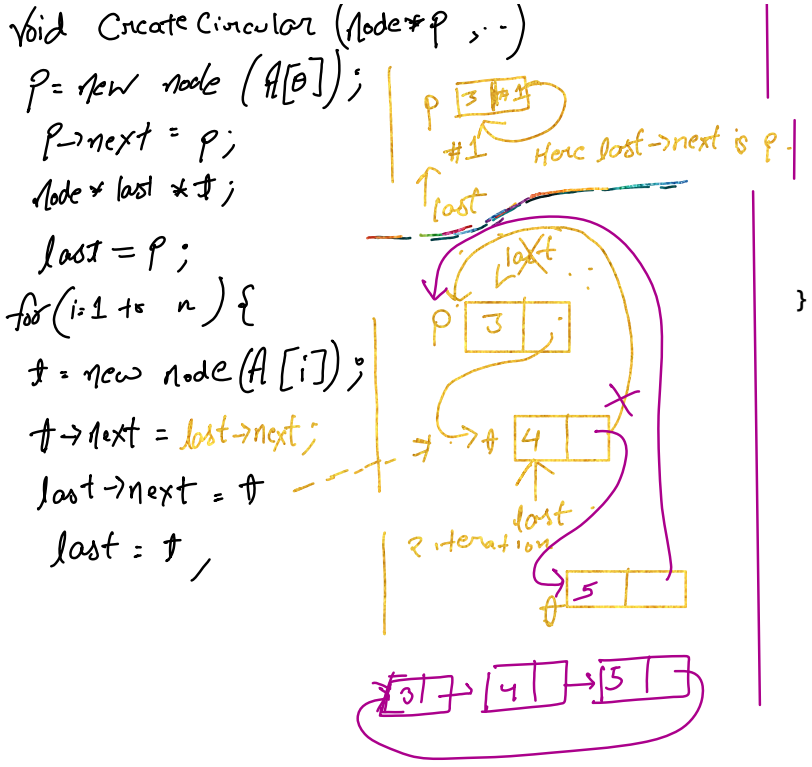
```
void display (node* p)
{
  node* temp;
  temp = p;
static int flag = 0;
  if(p!=temp || flag == 0)
  {
    flag = 1;
   cout<<p->data;
   display(p->next);
  }
      flag = 0;
}
```

Create Circular Linked List.

☆ Function of Creat.

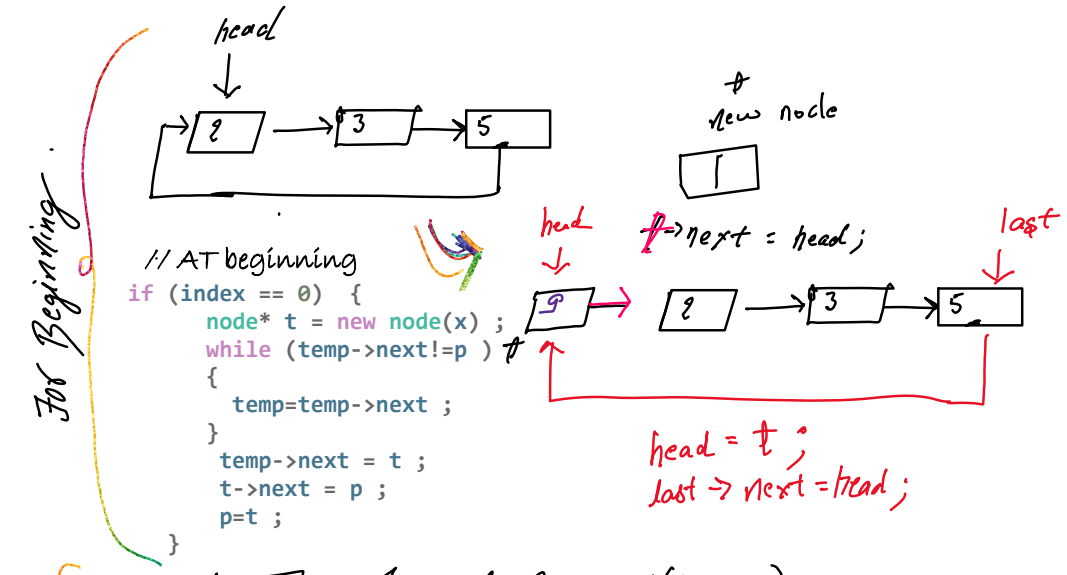
```
Void CreateCircular (node*p ,..)
p = new node (A[0]);
```

```
void createCricular(node* & p , int A[] , int n){
    p= new node(A[0]);
    p->next = p ;  // notable
    node* last = p ;
    node* t ;
```

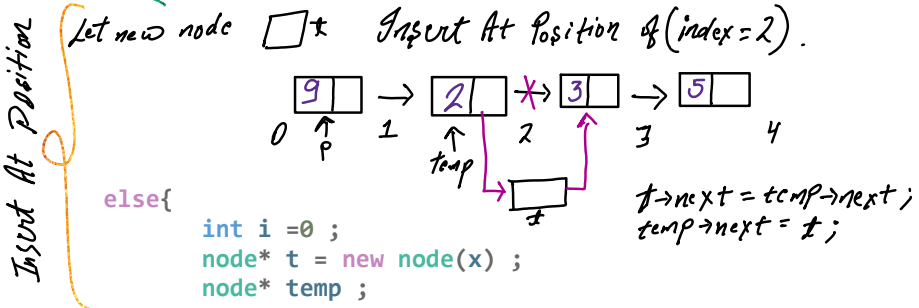



```
p->next = p ;  // notable
node* last = p ;
node* t ;
for (int i = 1; i < n; i++)
{
  t = new node(A[i]) ;
  t->next =  last->next ;  // notable
  last ->next  = t ;
  last = t ;
}
}
```

Insertion



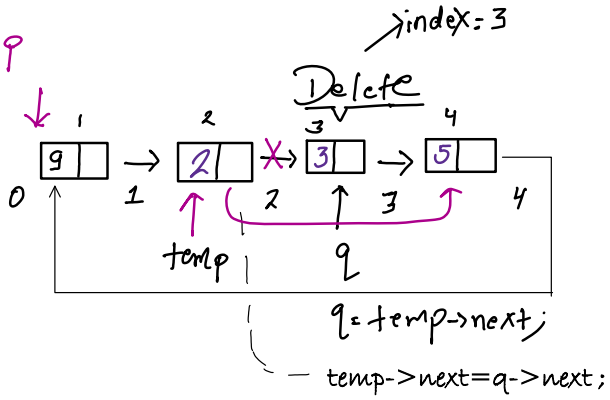

```
// AT beginning
if (index == 0)  {
      node* t = new node(x) ;
      while (temp->next!=p )
      {
        temp=temp->next ;
      }
       temp->next = t ;
       t->next = p ;
       p=t ;
   }
```




```
else{
      int i =0 ;
      node* t = new node(x) ;
      node* temp ;
      temp = p ;
       while ( i < index-1  ){
           temp=temp->next ;
          i++ ;
       }
       t->next = temp->next ;
       temp->next = t ;
```

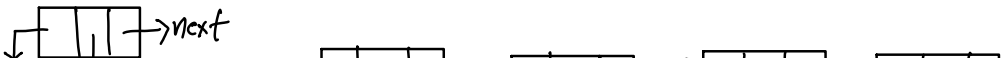
```
    }
}
```

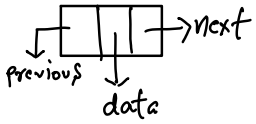
Delete At position (function)

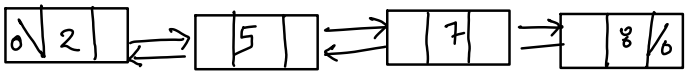



```
void Cridelete(node* &p, int index){
    node* temp = p;
    node* t = p;
    int x ;
    if(index<=0 || index > 4 ){
        return ;
    }
    if (index == 1)
    {
        while(temp->next!=p)temp=temp->next;
        x=p->data ;
        if(p==temp){
            free(p) ;
            p=NULL ;
        }
        else {
           p=p->next ;     //  temp->next = p->next ;
           temp->next = p ;  //  free(p);
           free(t);          // p= t-> next ;
        }
         cout<<x<<"\n";
    }
    else{
        for (int i = 0; i < index-2 ; i++)temp=temp->next ;
        node* q ;
        q=temp->next ; x=q->data ;
        temp->next = q->next ;
        free(q) ;
        cout<<x<<"\n";
    }
}
```

Doubly Linked List







So here we need to make Previous (Pre) node in Class.

Ex:

```
Class node {
  node * pre ;
  int data ;
  node * next ;
}
```

→ Format

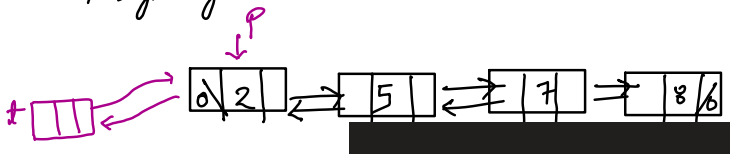
Create.

```
P = new node (A[0]);
temp = P;
P→Pre = NULL;
P→next = NULL;
for( 1 to n){
node * t = new node (A[i]);
    P→next = t;
    t→pre = P;
    P = P→next;
    t→next = NULL;
}
   P = temp;
```

```
void create(node* &p , int a[] , int n){
        p=new node(a[0]) ;
        node*temp = p ;
        p->next = NULL ;
        p->pre = NULL ;
        for ( int i = 1; i < n; i++)
        {
            node* t = new node(a[i]) ;
            p->next = t ;
            t->pre = p ;
            p = p->next ;
            // p=t ;
            t->next = NULL ;
        }
        p=temp ;
}
```

Insert

(i) At Beginning.




```
node * t=new node (x);
 t → Pre = NULL;
 t → next = P;
 P → Pre = t;
  P =t;
```

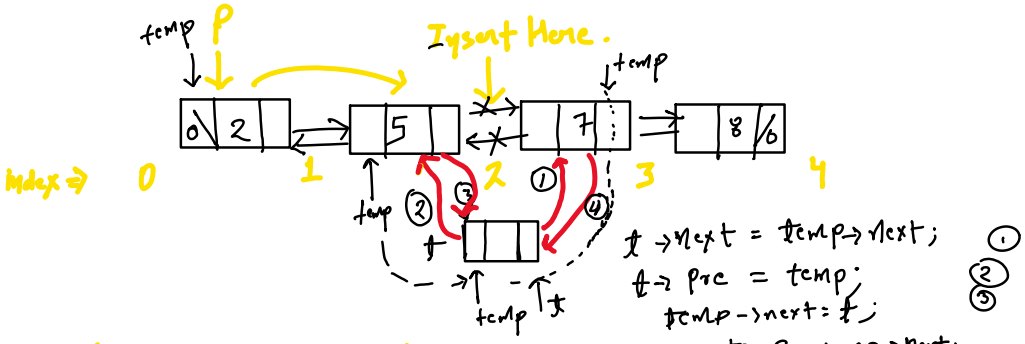
```
if (index == 0)
    {
        node *t = new
node(x);
        t->pre = NULL;
        t->next = p;
        p->pre = t;
        p = t;
```

p → pre = t;

p = t;

```
        t->next = p;
        p->pre = t;
        p = t;
    }
```

(ii) Insert At position



If u want put in 2nd index then we need to shift (p) one times. So iteration is (index - 1).

```
else
   {
        for (int i = 0; i < index - 1; i++)
            temp = temp->next;
        node *t = new node(x);
        t->next = temp->next;
        t->pre = temp;
        /*------>*/
        if (temp->next)
        {
            temp->next->pre = t; // best aproch
        }
        temp->next = t;
   }
```