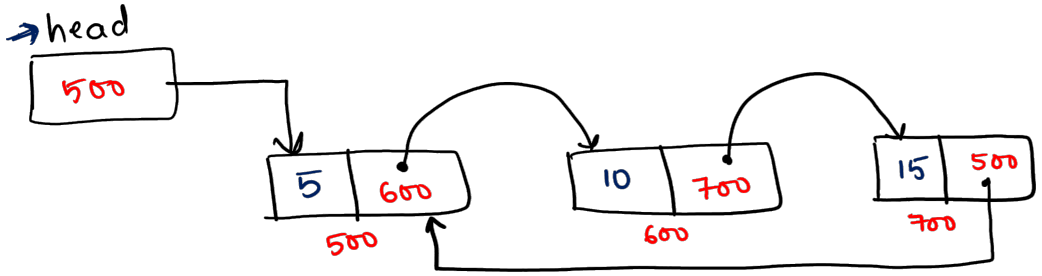
Circular Single Linked List :-



Node Structure :-

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

head Node :-

```
Struct Node *head = NULL;
```

Create a new Code :-

```
Struct Node *nn = (struct Node *) malloc (sizeof(struct Node));
```

Operations:

1) Insert
   a) at begin
   b) at end
   c) at given position

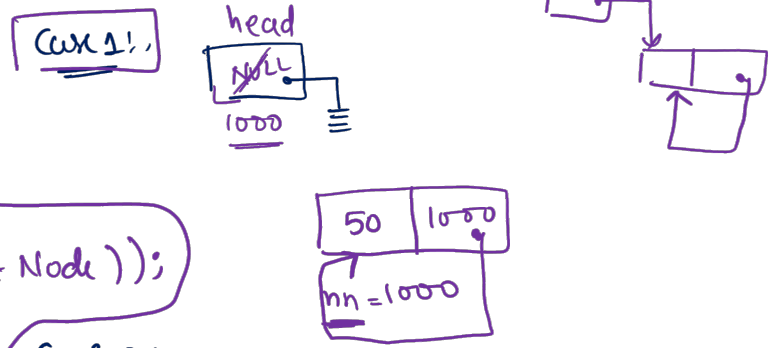
2) Deletion:
   a) at begin
   b) at end
   c) at given position

3) display / traversing.

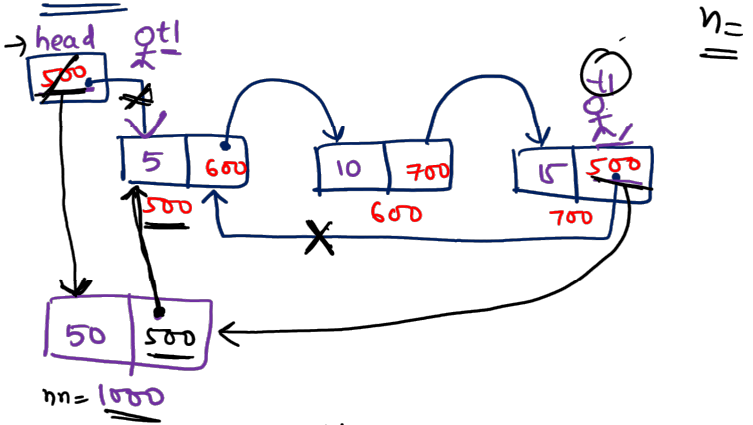
*4) search operation.

Insert at begin:-

```
Void insertatbegin(int item)
{
Struct Node *nn = (Struct Node *) malloc( sizeof(Struct Node));
   nn → data = item;
 if ( head == NULL)
  {
        nn → next = nn;
        head = nn;
  }
  else {
  Struct Node *tl = head;
while (tl → next != head)
 {   tl = tl → next;
 }
nn → next = head;
head = nn;
tl → next = nn;

n++;
}
}
```



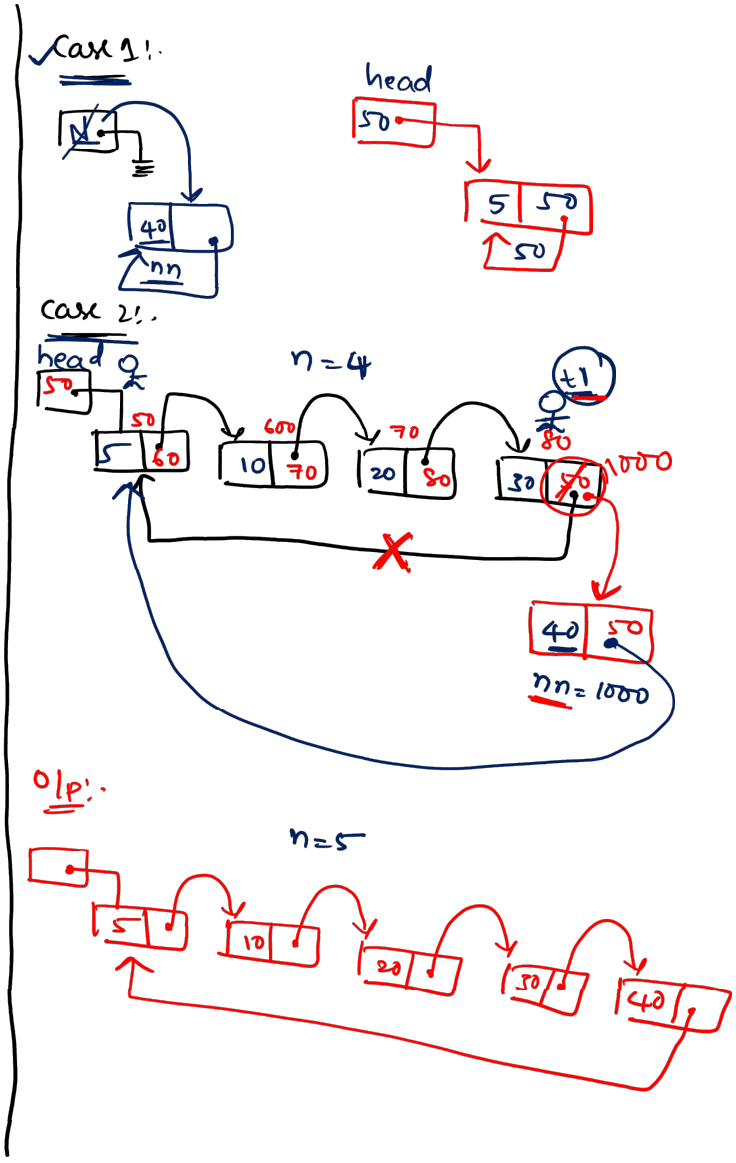
Case 2:-



Note:- if tl → next is "head" then "node" is last node

tl

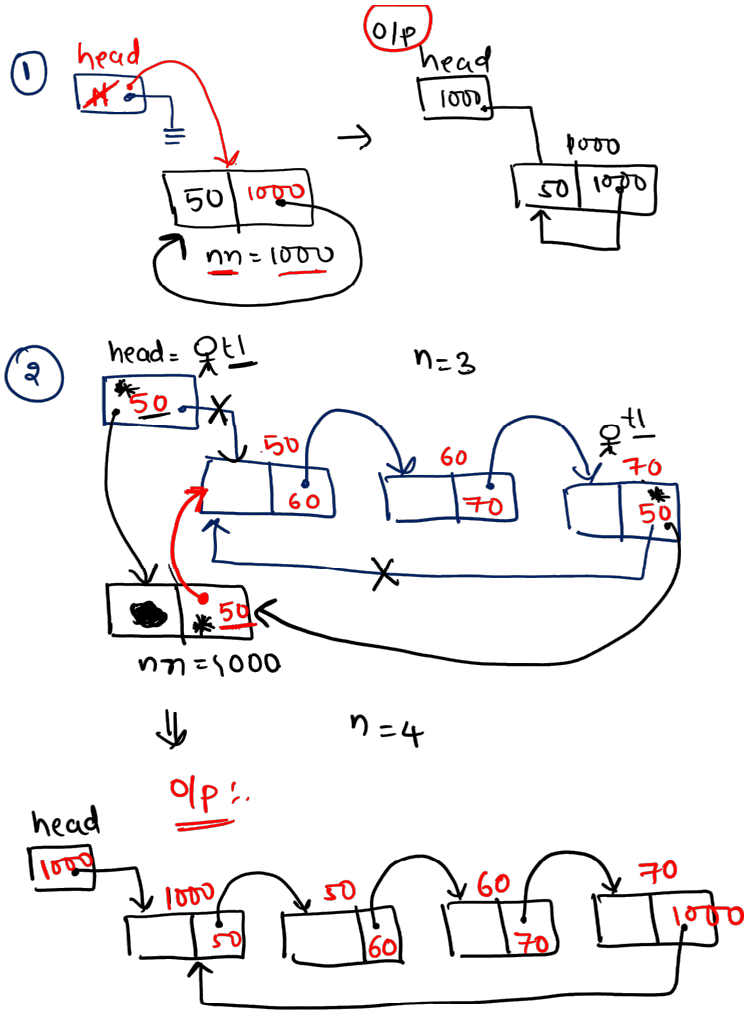
```
void insert at end(int item)
{
    struct Node *nn = (struct Node *) malloc( sizeof (struct Node));

        nn -> data = item;

    if (head == NULL)
    {
        nn -> next = nn;  // bcz only one element is present
        head = nn;
    }
    else
    { struct Node *tl = head;
    while (tl -> next != head)
      {
          tl = tl -> next;
      }
        nn -> next = head;
        tl -> next = nn;
} }   n++;
```



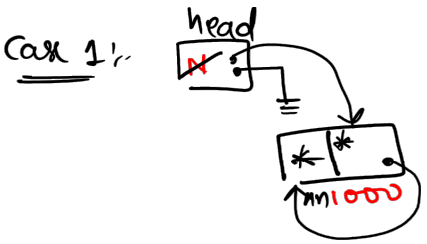
Insert into C.LL.

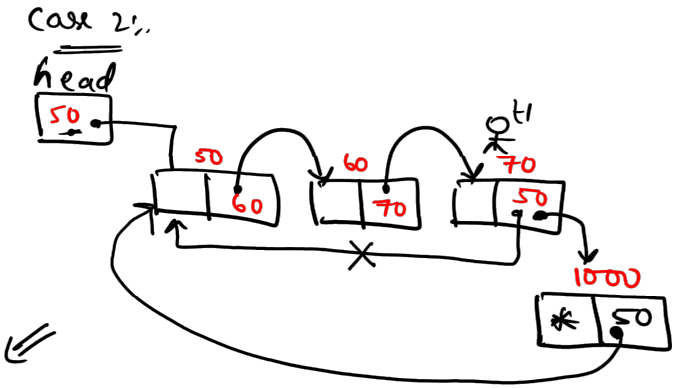
(1) Insert at begin:.

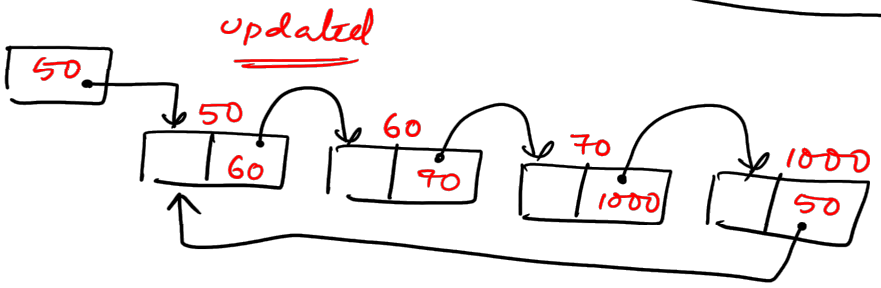
```
Void insert at begin(int item)
{
struct Node *nn = (Struct Node *) malloc( sizeof (Struct Node));
    nn->data = item;
   if( head ==NULL)
   {
       nn->next = nn;
        head = nn;
    }
   else
   {  Struct Node *tl = head;
      while(tl->next !=head)
       {
           tl=tl->next;
        }
      nn->next = head;
      tl->next = nn;
      head = nn ;
      n++;
   }
}
```



```
void insertatend(int item)
{
struct Node *nn=(struct Node *)malloc(sizeof(sN));
    nn->data=item;
    if(head==NULL)
    {
      nn->next=nn;
       head=nn;
    }
    else
    { struct Node *tl=head;
    while(tl->next!=head)
      {
      tl=tl->next;
      }
      nn->next=tl->next;
      tl->next=nn;
        n++;
      }}
```



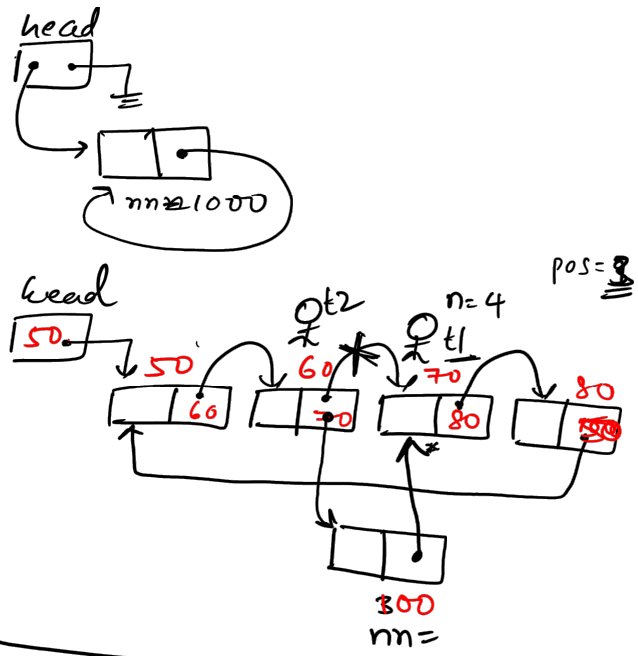




Void insert at given position (int item, int pos)

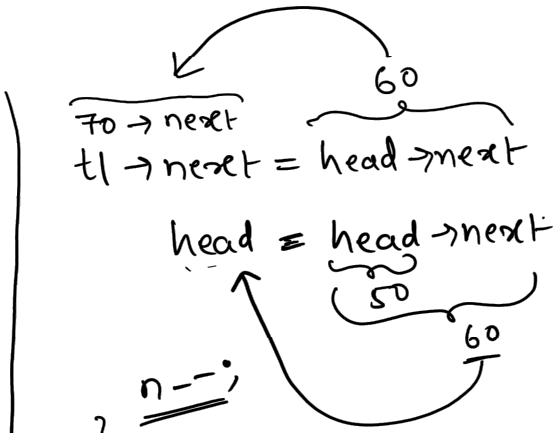
```
{
struct Node *nn = (struct Node *) malloc (sizeof(SN));
    nn->data = item;
if(head == NULL)
 {
  print("Actually list is Empty - I am adding at start position");
   nn->next = nn;
    head = nn;
 }
else
 {
  if(pos > n)
  {
   printf("Invalid pos");

  }
```

```
else
 {
 struct Node *t1 = head
 struct Node *t2;

 for(i=1; i<pos; i++)
 {   t2 = t1;
     t1 = t1->next;
 }
     nn->next = t1
     t2->next = nn;
        n++;
 } } }
```

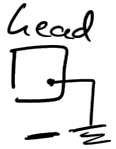


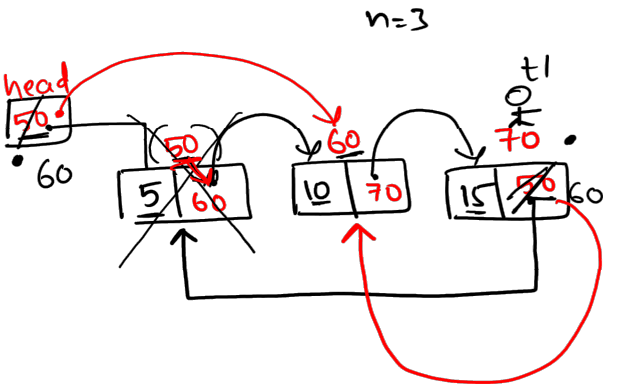
Deletion :- at begin

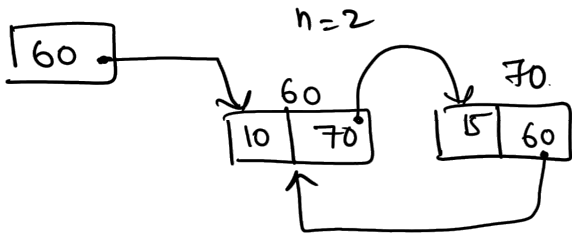
```
void deleteatBegin( )
{
   ie(head == NULL)
   {
     print("List is Empty");
   }
   else
   {
      struct Node *tl = head;
      while(tl->next != head)
        {
          tl = tl->next;
        }
```




```
}
```

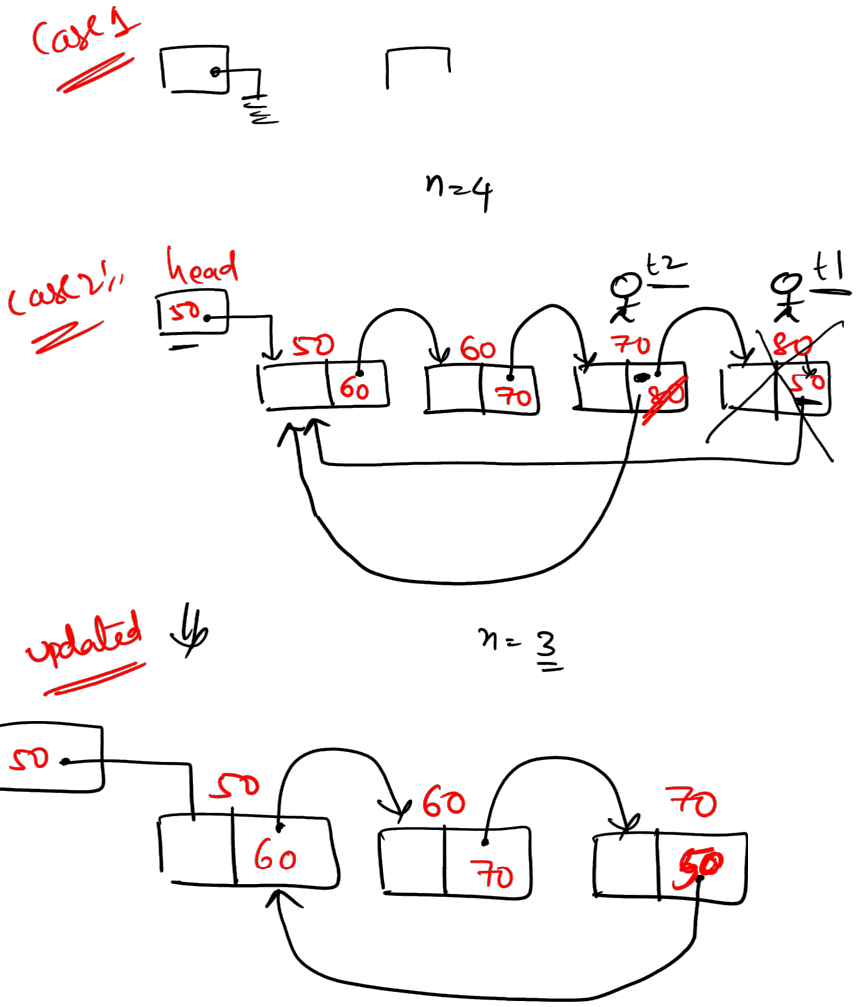




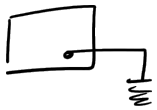


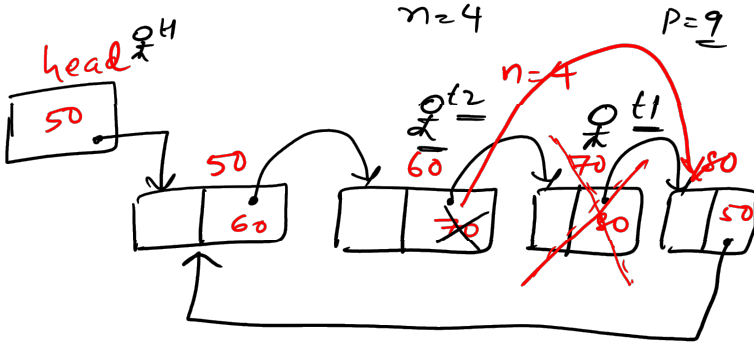
delete at end:

```
Void delete at end( )
{
  if(head == NULL)
  {
    print("List is Empty");
  }
  else{
    Struct Node *t1 = head;
    Struct Node *t2;
  while(t1->next != head)
    {
        t2 = t1;
        t1 = t1->next;
    }
    t2->next = head;
    n--;
} }
```



```
-Void deleteat given position (int pos)
{
    if(head = NULL)
    {
        print(" List is Empty ");
    }
    else
    {
        if ( pos>n )
        {
            printf(" Invalid deletion");
        }
        else
        {
            struct Node *t1=head;
            struct Node *t2;
            for(i=1; i<pos ; i++)
            {
                t2 =t1;
                t1= t1->next;
            }
            t2->next = t1->next;
            n--;
        }
    }
}
```

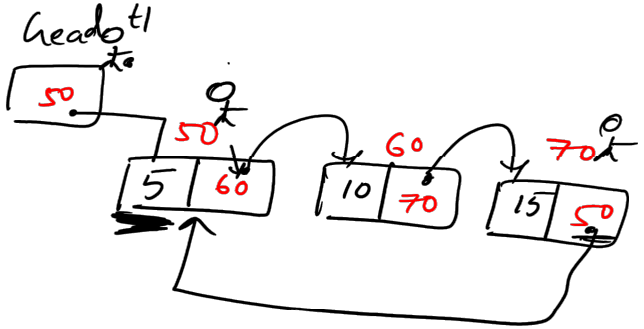




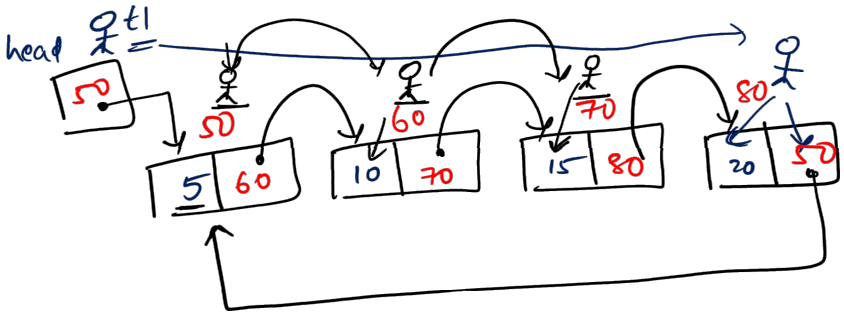
P = 3

n = 3

```
Void display( )
{
    if( head ==NULL)
    {
      printf(" List is empty");
    }
    else
    {
       Struct node *tl = head;
  while(tl != head)
     {
        printf(" %d", tl -> data);
        tl = tl -> next;
     }
}
```



```
void search(int se)
{
    int pos = 0;
    int status = 0;
    struct Node *tl = head;
    while (tl != head)
    {
        if (tl->data == se)
        {
            printf("found at %d", pos);
            status = 1;
        }
        tl = tl->next;
        pos++;
    }
    if (status == 0)
    {
        printf("element does n't found");
    }
}
```



se = 15

Go pos = 0

status = 0