```c

// INSERTION AND DELETION FROM SINGLY LINKED LIST
//calculate nth node from end in 2 pass
// calculate nth node from end in 1 pass::::::::::::: IMPORTANT::::::::::
// Reversing the Linked List
#include<stdio.h>
#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
};

struct node *head=NULL;
int V,C;

void InsertInB(int Value)// insert the node in the Beginning
{
    V=Value;
    struct node *p=(struct node *)malloc(sizeof(struct node));
    p->data=V;

    if(head==NULL)
    {
        head=p;
        p->next=NULL;
    }
    else
    {
        struct node *q=head;
        p->next=q;
        head=p;

    }
}

void InsertInE(int Value)   // insert the node in the End
{
    V=Value;
    struct node *p=(struct node *)malloc(sizeof(struct node));
    p->data=V;

    if(head==NULL)
    {
        head=p;
        p->next=NULL;
    }
    struct node *q=head;

    while(q->next!=NULL)
    {
        q=q->next;
    }
    q->next=p;
    p->next=NULL;
}

int LengthOfLL() // traverse LL to Calculate the length of LL
{
    C=0;
    struct node *p=head;
    while(p!=NULL)
    {
            p=p->next;
            C++;
    }
    return C;
}

void InsertInM(int Value,int index) // insert the node in the middle of LL at particular Index.
{
    V=Value;
    struct node *p=(struct node *)malloc(sizeof(struct node));
    p->data=V;

    struct node *q=NULL,*r=NULL;
    if(index>LengthOfLL())
    {
        printf("incorrect Index Value");
    }

    else
    {
        int i;
        q=head;
        for(i=1;i<=index-1;i++)
        {
            r=q;
            q=q->next;
        }

        p->next=q;
        r->next=p;
    }

}

void DeleteFrmB() // delete the First Node in LL.
{
    struct node *p=head;
    head=head->next;
    free(p);
}

void DeleteFrmE() // delete the Last Node in LL.
{
    struct node *p=head;
    while(p->next->next!=NULL)
    {
        p=p->next;

    }
    free(p->next);
    p->next=NULL;
}

void DeleteFrmM(int index)  // Delete the node from the middle at given index.
{
    int i;
    struct node *p=NULL,*q=NULL;
    p=head;
    for(i=1;i<=index-1;i++)
    {
        q=p;
        p=p->next;
    }

    q->next=p->next;
    free(p);
}
```

```c

void CalculateNthFromE(int index) // it will take two pass of scan to calculate..
{
    if(index>LengthOfLL()+1)
        printf("out of bound Index");
    else
    {
        int i;
        struct node *p=head;
        int N=LengthOfLL()-index;
        for (i=1;i<=N;i++)
        {
            p=p->next;
        }
        printf("%d th node from last is %d\n",index,p->data);
    }
}

void CalculateNthFromEin1Pass(int index)// Calculate Nth node from last in single pass :::::IMPORTANT::::::::::::
{
    struct node *p=head,*q=head;
    int i;
    for (i=1;i<index;i++)
    {
        p=p->next;
    }

    while (p->next!=NULL)
    {
        p=p->next;
        q=q->next;
    }
    printf("\n\n%dth node from end in single pass %d\n",index,q->data);
}

void reverseLL() // Reverse the Linked List/////////////////////////////////////////////
{
    struct node *p=head,*q=NULL,*r=head;

        while(r!=NULL)
        {
           r=r->next;
           p->next=q;
           q=p;
           p=r;
        }

    head=q;


}
////// print the LL from end recursively/////////////////////////////////////////
void PrintListFromEnd(struct node *head)
{
    if(!head)
        return;
    PrintListFromEnd(head->next);
    printf("%d",head->data);
    printf("--->");
}
/////////// check given Linked List is Even or Odd in length//////////////////////////////
int CheckEvenOdd()
{
    struct node *p=head;
    while(p&&p->next)
    {
        p=p->next->next;

    }

    if(p==NULL)
        printf("\n Even Length");
    else
        printf("\n\n Odd length");




}

void displayLL() // display the complete LL created.
{
    struct node *dp=head;
    while(dp!=NULL)
    {
        printf("%d",dp->data);
        printf("--->");
        dp=dp->next;
    }

}

void main()
{
    InsertInB(20);
    displayLL();
    printf("\n\n");
    InsertInB(30);
    InsertInB(40);
    displayLL();
    printf("\n\n");
    InsertInE(10);
    displayLL();
    printf("\n\n");
    InsertInM(500,3);displayLL();
    printf("\n\n");
    printf("Length of Current LL is %d\n",LengthOfLL());

   CalculateNthFromE(5);
   CalculateNthFromEin1Pass(4);

    DeleteFrmM(3);
    displayLL();
    printf("\n\n");
    DeleteFrmB();
    displayLL();
    printf("\n\n");
    DeleteFrmE();
    displayLL();
    printf("\n\n");
    InsertInB(12);InsertInB(32);InsertInB(21);InsertInB(65);
    displayLL();
    reverseLL();
    printf("\n\nReverseing the linked List\n\n");
    displayLL();
    printf("\n\n");
    //PrintListFromEnd(head);
    CheckEvenOdd();

```

```
}
```