1. struct Node

{

int data;

struct Node \*next;

}

InsertNth(Node \*head, int data, int position)

{

Node \* newnode = (Node \*)malloc(sizeof(Node));

newnode->data=data;

Node \*temp=head,\*prev;

if(position==0){

newnode->next=head;

head=newnode;

return head;

}

else{

while(position--){

prev=temp;

temp=temp->next;

}

prev->next=newnode;

newnode->next=temp;

return head;

}

}

**2.** void delete\_beg()

{

    if(start==NULL)

    {

        printf("The list is empty!!");

    }

    else

    {

        q=start;

        start=start->next;

        printf("Deleted element is %d",q->data);

        free(q);

    }

}

**3. Function to delete a node from the end of a linked list**

void delete\_end()

{

if(header->link == NULL)

{

printf("\nEmpty Linked List. Deletion not possible.\n");

}

else

{

//Traverse to the end of the list.

ptr = header;

while(ptr->link != NULL)

{

ptr1 = ptr;

ptr = ptr->link;

}

ptr1->link = ptr->link;

free(ptr);

printf("\nNode deleted from the end.\n");

}

}

**4.** Because the iterator returned from end does not denote an element, it may not be incremented or dereferenced. The beg and end are unsigned values.

The first one is used for affine types like pointers and other random-access iterators, which can be subtracted to give a distance, but not added together.

There are maybe an asymmetric range, [begin, off-the-end) or half-open range.

**5.**

int ternary\_search(int l,int r, int x)

{

if(r>=l)

{

int mid1 = l + (r-l)/3;

int mid2 = r - (r-l)/3;

if(ar[mid1] == x)

return mid1;

if(ar[mid2] == x)

return mid2;

if(x<ar[mid1])

return ternary\_search(l,mid1-1,x);

else if(x>ar[mid2])

return ternary\_search(mid2+1,r,x);

else

return ternary\_search(mid1+1,mid2-1,x);

}

return -1;

}