```c
#include<stdio.h>
#include<stdlib.h>

struct node
{
    int data ;
    struct node *next ;
}*start=NULL,*previous=NULL;
typedef struct node * nodeptr ;

void create()
{
    int n;
    printf("How many Nodes to create at start : ");
    scanf("%d",&n);
 while(n>0)
 {
    nodeptr new_node,current;
    new_node=(nodeptr)malloc(sizeof(struct node));
    printf("Enter the data : ");
    scanf("%d",&new_node->data);
    new_node->next=NULL;

  if(start==NULL)
  {
    start=new_node;
    current=new_node;
  }
  else
  {
    current->next=new_node;
    current=new_node;
  }
  n--;
 }
}

nodeptr getnode(int item)
{
    nodeptr p ;
    p=(nodeptr) malloc(sizeof(struct node));
    p->data=item ;
    return p ;
}

void display()
{
    nodeptr p;
    p=start;
    while (p!=NULL)
    {
        printf("%d %p\n",p->data,p);
        p=p->next;
    }
}


void insertFirst(int item)
{
    nodeptr p=getnode(item);
    p->next=start;
    start=p;
}

void insertLast(int item)
{
    if (start==NULL)
        insertFirst(item);
    else
    {
        nodeptr p,q ;
```

```
        p=getnode(item);
        q=start ;
        while(q->next!=NULL)
            q=q->next;
        q->next=p;
        p->next=NULL;
    }
}

void insertAfter(int item,int pre)
{
    nodeptr q,p=getnode(item);
    q=start;
    while(q->data!=pre && q!=NULL)
        q=q->next;
    if(q==NULL)
        printf("Previous not found");
    else
    {
        p->next=q->next;
        q->next=p ;
    }
}

void insertSorted(int item)
{
    nodeptr p,r;
    p=start;
    r=NULL;

    if(start==NULL)
        insertFirst(item);
    else
    {
        while(p->data<=item && p!=NULL)
        {
            r=p;
            p=p->next;
        }

        if (r==NULL)
            insertFirst(item);
        else
        {
            previous=r;
            insertAfter(item,previous->data);
        }
    }
}

int deleteFirst()
{
    int x;
    nodeptr p;

    if(start==NULL)
    {
        printf("Linked List is Empty");
        return -32768 ;
    }

    p=start;
    start=start->next ;
    x=p->data;
    free(p);
    return x;
}

int deleteAfter(int pre)
{
    int x;
```

```
    nodeptr p,q;
    q=start;
    while(q->data!=pre && q!=NULL)
        q=q->next;

    if (start==NULL)
    {
        printf("Linked List is empty");
        return -32768 ;
    }
    else if (q==NULL || q -> next==NULL)
    {
        printf("Previous is NULL or no node after previous to delete");
        return -32768 ;
    }
    else
    {
        p=q->next;
        x=p->data;
        q->next=q->next->next ;
        free(p);
        return x ;
    }
}

int deleteLast()
{
    int x;
    nodeptr p,q;
    p=start;
    q=NULL;
    while(p->next!=NULL)
    {
        q=p;
        p=p->next;
    }
    if(p==NULL)
    {
        printf("Linked List is empty") ;
        return -32768 ;
    }
    else if(q==NULL)
        x=deleteFirst();
    else
    {
        previous=q;
        x=deleteAfter(previous->data);
    }
    return x;
}

int deleteVX(int item)
{
    int x;
    nodeptr p,q;
    p=start;
    q=NULL;

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

    if (p==NULL)
    {
        if(q==NULL)
            printf("Linked List is Empty");
        else
            printf("Item not Found");
        return -32768 ;
```

```
    }
    else
    {
        if (q==NULL)
            x=deleteFirst();
        else
        {
            previous=q;
            x=deleteAfter(previous->data);
        }
        return x;
    }
}

void main()
{
    int n,x,y ;
    create();
    do
    {
        system("cls");
        display();
        printf("\n1.Insert First\n2.Insert Last\n3.Insert After\n4.Insert Sorted\n5.Delete
        First\n6.Delete Last\n7.Delete After\n8.Delete With Value X\n9.Display Linked
        List\n");
        scanf("%d",&n) ;
        if ((n>0 && n<5) || n==8)
        {
            printf("Insert item ");
            scanf("%d",&x) ;
        }
        if (n==3 || n==7)
        {
            printf("Enter Previous Value ");
            scanf("%d",&y);
        }

        switch(n)
        {
            case 1 :insertFirst(x);
                    break;
            case 2 :insertLast(x);
                    break;
            case 3 :insertAfter(x,y);
                    break;
            case 4 :insertSorted(x);
                    break;
            case 5 :deleteFirst();
                    break;
            case 6 :deleteLast();
                    break;
            case 7 :deleteAfter(y);
                    break;
            case 8 :deleteVX(x);
                    break;
            case 9 :break;
        }
    }while(n>0 && n<10) ;
}
```