```
/*
Title- Doubly Linked List Operations
Author- Bhakare Mahesh Santosh
ID- 492
Batch- TechnOrbit(PPA-8)
*/

#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node* prev;
    struct node* next;
};
int CountNode(struct node*);
struct node* CreateNode();
void CreateLinkedList(struct node**,struct node**);
void DisplayLinkedList(struct node*);
void ReverseDisplay(struct node*);
void InsertAtFirst(struct node**,struct node**);
void InsertAtLast(struct node**,struct node**);
void InsertAtPosition(struct node**,struct node**);
void DeleteAtFirst(struct node**,struct node**);
void DeleteAtLast(struct node**,struct node**);
void DeleteAtPosition(struct node**,struct node**);

void main()
{
    struct node *first = NULL, *last = NULL;
    int choice;
    do
    {
        printf("\n------------------------------------ ************************
-------------------------------------------\n");
        printf("\n1) Create Linked List\n2) Display Linked List\n3) Reversed Linked 
List\n4) Insert at First\n5) Insert At Last\n6) Insert At Position\n7) Delete At 
First\n8) Delete At Last\n9) Delete At Position\n0) Exit\nEnter Your Choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: CreateLinkedList(&first,&last);
                    break;
            case 2: DisplayLinkedList(first);
                    break;
            case 3: ReverseDisplay(last);
                    break;
            case 4: InsertAtFirst(&first,&last);
                    break;
            case 5: InsertAtLast(&first,&last);
                    break;
            case 6: InsertAtPosition(&first,&last);
                    break;
            case 7: DeleteAtFirst(&first,&last);
                    break;
            case 8: DeleteAtLast(&first,&last);
                    break;
            case 9: DeleteAtPosition(&first,&last);
                    break;
        }
    }while(choice!=0);
}
int CountNode(struct node* head)
```

```c
{
    int count = 0;
    while(head != NULL)
    {
        count++;
        head = head->next;
    }
    return count;
}


struct node* CreateNode()
{
    struct node* newnode =  NULL;
    newnode = (struct node*)malloc(sizeof(struct node));
    if(newnode == NULL)
    {
        printf("Memory not allocated\n");
    }
    else
    {
        printf("Enter the data: ");
        scanf("%d",&(newnode->data));
        newnode->prev = NULL;
        newnode->next = NULL;
    }
    return newnode;
}



void CreateLinkedList(struct node** first, struct node** last)
{
    struct node* newnode = NULL;
    newnode = CreateNode();
    if(*first == NULL)
    {
        *first = *last = newnode;
    }
    else
    {
        newnode->prev = *last;
        (*last)->next = newnode;
        *last = newnode;
    }
}


void DisplayLinkedList(struct node* head)
{
    printf("Linked List In Forward Order: ");
    while(head != NULL)
    {
        printf(" -> %d",head->data);
        head = head->next;
    }
}


void ReverseDisplay(struct node* head)
{
    printf("Linked List In Backward Order: ");
    while(head != NULL)
    {
        printf(" -> %d",head->data);
        head = head->prev;
    }
}

void InsertAtFirst(struct node** first, struct node** last)
{
    struct node* newnode = NULL;
    newnode =CreateNode();
    if(*first == NULL)
    {
        *first = *last = newnode;
    }
    else
    {
        newnode->next = *first;
        (*first)->prev = newnode;
        *first = newnode;
    }

}


void InsertAtLast(struct node** first, struct node** last)
{
    CreateLinkedList(first, last);
}

void InsertAtPosition(struct node** first, struct node** last)
{
    struct node* tempnode = *first;
    struct node* newnode = NULL;
    int n,pos,i;
    n= CountNode(*first);
    printf("enter the position where you want to insert a new node: ");
    scanf("%d",&pos);
    if(pos == 1)
    {
        InsertAtFirst(first, last);
    }
    else if(pos == n+1)
    {
        InsertAtLast(first, last);
    }
    else if(pos < 1 || pos > n+1)
    {
        printf("Invalid Position....Please Enter Position Again...\n");
        InsertAtPosition(first, last);
    }
    else if(pos > 1 && pos < n+1)
    {
        newnode = CreateNode();
        for(i = 1; i<pos;i++)
        {
            tempnode= tempnode->next;
        }
        newnode->next = tempnode;
        newnode->prev = tempnode->prev;
        tempnode->prev->next = newnode;
        tempnode->prev = newnode;
    }
}

void DeleteAtFirst(struct node** first, struct node** last)
{
    struct node* tempnode = *first;
    if(*first == NULL)
    {
        printf("Linked List not Available....\n");
    }
    else if((*first)->next == NULL)
    {
        free(*first);
        *first = *last = NULL;
    }
    else
    {
        *first= (*first)->next;
        (*first)->prev = NULL;
        free(tempnode);
        tempnode = NULL;
    }
}
void DeleteAtLast(struct node** first , struct node** last)
{
    if(*first == NULL)
    {
        printf("Linked List not Available...\n");
    }
    else if((*first)->next == NULL)
    {
        free(*first);
        *first = *last = NULL;
    }
    else
    {
        *last = (*last)->prev;
        free((*last)->next);
        (*last)->next = NULL;
    }
}

void DeleteAtPosition(struct node** first, struct node** last)
{
    struct node* tempnode = *first;
    int n,pos,i;
    printf("Enter the position from where you want to delete element: ");
    scanf("%d",&pos);
    n = CountNode(*first);
    if(pos == 1)
    {
        DeleteAtFirst(first, last);
    }
    else if(pos == n)
    {
        DeleteAtLast(first, last);
    }
    else if(pos > 1 && pos < n)
    {
        for( i = 1; i<pos;i++)
        {
            tempnode = tempnode->next;
        }
        tempnode->next->prev = tempnode->prev;
        tempnode->prev->next = tempnode->next;
        free(tempnode);
        tempnode = NULL;
    }
    else if(pos < 1 || pos > n)
    {
        printf("Please enter the valid position...\n");
        DeleteAtPosition(first, last);
    }
}
```