```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head=NULL,*tail=NULL;
void create(void);
void display(void);
void ins_at_beg(int);
void ins_at_end(int);
void ins_at_pos(int);
void del_at_beg(void);
void del_at_end(void);
void del_at_pos(void);
int length(void);
void reverse(void);
void main()
{
    int choice,info;
    while(1)
    {
        printf("1.Create\n");
        printf("2.Insert at beginning\n");
        printf("3.Insert at ending\n");
        printf("4.Insert at specific position\n");
        printf("5.Delete a node at beginning\n");
        printf("6.Delete a node at ending\n");
        printf("7.Delete a node at specific position\n");
        printf("8.Display\n");
        printf("9.Reverse\n");
        printf("10.Exit\n");
        printf("Enter your choice ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: create();
                    break;
            case 2: printf("Enter your data ");
                    scanf("%d",&info);
                    ins_at_beg(info);
                    break;
            case 3: printf("Enter your data ");
                    scanf("%d",&info);
                    ins_at_end(info);
                    break;
            case 4: printf("Enter your data ");
                    scanf("%d",&info);
                    ins_at_pos(info);
                    break;
            case 5: del_at_beg();
                    break;
            case 6: del_at_end();
                    break;
            case 7: del_at_pos();
                    break;
            case 8: display();
                    break;
            case 9: reverse();
                    break;
            case 10: exit(0);
            default: printf("Invalid input\n");
        }
    }
}
void create()
{
    struct node *newnode;
    int info,ch=1;
    while(ch)
    {
        newnode=(struct node *)malloc(sizeof(struct node));
        if(newnode==NULL)
        {
            printf("Malloc error\n");
            return;
        }
        printf("Enter your data ");
        scanf("%d",&info);
        newnode->data=info;
        newnode->next=NULL;
        newnode->prev=NULL;
        if(head==NULL)
        {
            head=tail=newnode;
        }
        else
        {
            newnode->prev=tail;
            tail->next=newnode;
            tail=newnode;
        }
        printf("Do you want to continue ");
        scanf("%d",&ch);
    }
}
void ins_at_beg(int info)
{
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("malloc error\n");
        return;
    }
    newnode->data=info;
    newnode->next=NULL;
    newnode->prev=NULL;
    if(head==NULL)
    {
        head=tail=newnode;
    }
    else
    {
```

```c
        head->prev=newnode;
        newnode->next=head;
        head=newnode;
    }
}
void ins_at_end(int info)
{
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Malloc Error\n");
        return;
    }
    newnode->data=info;
    newnode->next=NULL;
    newnode->prev=NULL;
    if(head==NULL)
    {
        head=tail=newnode;
    }
    else
    {
        tail->next=newnode;
        newnode->prev=tail;
        tail=newnode;
    }
}
void ins_at_pos(int info)
{
    struct node *newnode,*temp;
    temp=head;
    int pos,i=1,l;
    printf("Enter your position ");
    scanf("%d",&pos);
    l=length();
    if(pos<=0||pos>l)
    {
        printf("Invalid Position\n");
        return;
    }
    newnode=(struct node *)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Malloc Error\n");
        return;
    }
    newnode->data=info;
    newnode->prev=NULL;
    newnode->next=NULL;
    while(i<pos-1)
    {
        temp=temp->next;
        i++;
    }
    newnode->prev=temp->next->prev;
    newnode->next=temp->next;
    temp->next=newnode;
    temp->next->prev=newnode;
}
void del_at_beg()
{
    struct node *temp;
    temp=head;
    if(head==NULL)
    {
        printf("List is empty\n");
        return;
    }
    head=head->next;
    head->prev=NULL;
    free(temp);
}
void del_at_end()
{
    struct node *temp;
    temp=tail;
    if(head==NULL)
    {
        printf("List is empty\n");
        return;
    }
    tail->prev->next=NULL;
    tail=tail->prev;
    free(temp);
}
void del_at_pos()
{
    struct node *temp;
    temp=head;
    int pos,i=1,l;
    printf("Enter your position ");
    scanf("%d",&pos);
    l=length();
    if(pos<=0||pos>l)
    {
        printf("Invalid Position\n");
        return;
    }
    while(i<pos)
    {
        temp=temp->next;
        i++;
    }
    temp->prev->next=temp->next;
    temp->next->prev=temp->prev;
    free(temp);
}
void display()
{
    struct node *temp;
    temp=head;
    if(temp==NULL)
    {
        printf("List is empty\n");
        return;
    }
    while(temp!=NULL)
    {
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
void reverse()
{
    struct node *previous,*current,*nextnode;
    previous=NULL;
    current=nextnode=head;
    while(nextnode!=NULL)
    {
        nextnode=current->next;
        current->next=previous;
        current->prev=nextnode;
        previous=current;
        current=nextnode;
    }
    tail=head;
    head=previous;
}
int length()
{
    struct node *temp;
    int count=0;
    temp=head;
    if(head==NULL)
    {
        printf("List is empty\n");
        return -1;
    }
    while(temp!=NULL)
    {
        count++;
        temp=temp->next;
    }
    return count;
}
```