```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	struct node *prev;
	int data;
	struct node *next;
};
struct node *createnode(int data)
{
	struct node *newnode=malloc(sizeof(struct node));
	newnode->prev=NULL;
	newnode->data=data;
	newnode->next=NULL;
	return newnode;
}
struct node *addToBeginning(struct node *head,int data)
{
	struct node *newnode=createnode(data);
	if(head!=NULL)
	{
		head->prev=head;
	}
	newnode->next=head;
	return newnode;
}
struct node *addToEnd(struct node *head,int data)
{
	struct node *newnode=createnode(data);
	if(head==NULL)
	{
		return newnode;
	}
	struct node *temp=head;
	while(temp->next!=NULL)
	{
		temp=temp->next;
	}
	temp->next=newnode;
	newnode->prev=temp;
	return head;
}
struct node *addToMiddle(struct node *head,int pos,int data)
{
	if(head==NULL||pos<=0)
	{
		printf("Invalid position\n");
		return head;
	}
	struct node *newnode=createnode(data);
```

```
        struct node *temp=head;
        while(pos>1 && temp->next!=NULL)
        {
                temp=temp->next;
                pos--;
        }
        newnode->next=temp->next;
        newnode->prev=temp;
        if(temp->next!=NULL)
        {
                temp->next->prev=newnode;
        }
 temp->next=newnode;
        return head;
}
struct node *deleteFromBeginning(struct node *head)
{
        if(head==NULL)
        {
                printf("List is empty\n");
                return NULL;
        }
        struct node *temp=head;
        head=head->next;
        if(head!=NULL)
        {
                head->prev=NULL;
        }
        free(temp);
        return head;
}
struct node *deleteFromEnd(struct node *head)
{
        if(head==NULL)
        {
                printf("List is empty\n");
                return NULL;
        }
        struct node *temp=head;
        while(temp->next!=NULL)
        {
                temp=temp->next;
        }
        if(temp->prev!=NULL)
        {
                temp->prev->next=NULL;
        }
        free(temp);
        return head;
}
```

```
struct node *deleteFromMiddle(struct node *head,int pos)
{
	if(head==NULL)
	{
		printf("List is empty\n");
		return NULL;
	}
	struct node *temp=head;
	while(pos>1 && temp->next!=NULL)
	{
		temp=temp->next;
		pos--;
	}
	if(temp==head)
	{
		head=deleteFromBeginning(head);
	}
	else if(temp->next==NULL)
	{
		head=deleteFromEnd(head);
	}
	else
 {
		temp->prev->next=temp->next;
		temp->next->prev=temp->prev;
		free(temp);
	}
	return head;
}
void printList(struct node *head)
{
	struct node *temp=head;
	while(temp!=NULL)
	{
		printf("%d",temp->data);
		temp=temp->next;
	}
	printf("NULL\n");
}
struct node *findElement(struct node *head,int key)
{
	struct node *current=head;
	while(current!=NULL)
	{
		if(current!=NULL)
		{
			printf("Element %d found in the list\n",key);
			return current;
		}
		current=current->next;
```

```
        }
        printf("Element not found");
        return NULL;
}
int main()
{
        struct node *head=NULL;
        int choice,data,pos;
        printf("\n1.Add to beginning");
        printf("\n2.Add to end");
        printf("\n3.Add to middle");
        printf("\n4.Delete from beginning");
        printf("\n5.Delete from end");
        printf("\n6.Delete from middle");
        printf("\n7.Search element");
        printf("\n8.Dispaly");
        printf("\n9.Exit");
        while(1)
        {
                printf("\nEnter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                {
                                printf("Enter data: ");
                                scanf("%d",&data);
                                head=addToBeginning(head,data);
                                break;
                                }
                        case 2:
                                {
 {
                                printf("Enter data: ");
                                scanf("%d",&data);
                                head=addToEnd(head,data);
                                break;
                                }
                        case 3:
                                {
                                printf("Enter position: ");
                                scanf("%d",&pos);
                                printf("\nEnter data: ");
                                scanf("%d",&data);
                                head=addToMiddle(head,pos,data);
                                break;
                                }
                        case 4:
                                {
                                head=deleteFromBeginning(head);
```

```
                    break;
                    }
                case 5:
                    {
                    head=deleteFromEnd(head);
                    break;
                    }
                case 6:
                    {
                    printf("Enter position: ");
                    scanf("%d",&pos);
                    head=deleteFromMiddle(head,pos);
                    break;
                    }
                case 7:
                    {
                    printf("Enter element: ");
                    scanf("%d",&data);
                    head=findElement(head,data);
                    break;
                    }
                case 8:
                    {
                    printf("List:");
                    printList(head);
                    break;
                    }
                case 9:
                    {
                    exit(0);
                    }
                default:
                    {
                    printf("Invalid choice\n");
                    }
            }
            return 0;
        }
}
```