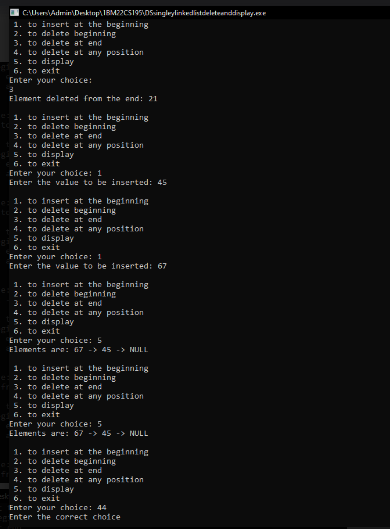
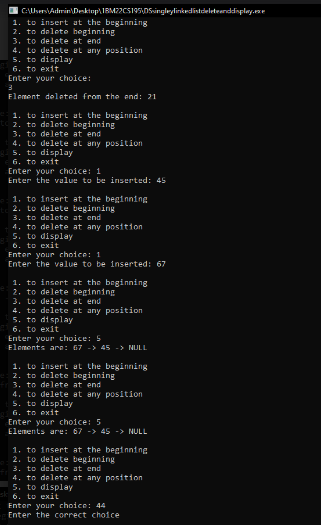
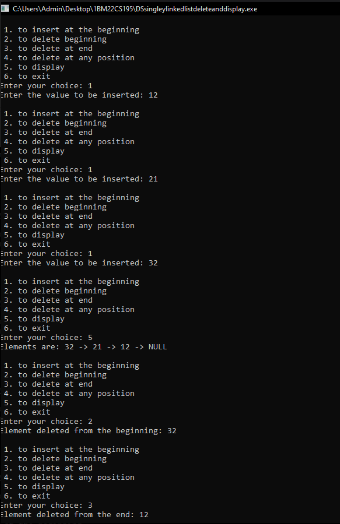
Lab 5[18-01-24]

include<stdio.h>  
#include<stdlib.h>  
struct node {  
    int data;  
    struct node \*next;  
};  
struct node \*head = NULL;  
void display() {  
    printf("Elements are: ");  
    struct node \*ptr = head;  
    while (ptr != NULL) {  
        printf("%d -> ", ptr->data);  
        ptr = ptr->next;  
    }  
    printf("NULL\n");  
}  
void insert\_begin() {  
    struct node \*temp = (struct node\*)malloc(sizeof(struct node));  
    printf("Enter the value to be inserted: ");  
    scanf("%d", &temp->data);  
    temp->next = head;  
    head = temp;  
}  
void delete\_begin() {  
    if (head == NULL) {  
        printf("List is empty. Deletion not possible.\n");  
        return;  
    }  
    struct node \*temp = head;  
    head = head->next;  
    printf("Element deleted from the beginning: %d\n", temp->data);  
    free(temp);  
}  
void delete\_end() {  
    if (head == NULL) {  
        printf("List is empty. Deletion not possible.\n");  
        return;  
    }  
    struct node \*temp, \*prev;  
    temp = head;  
    while (temp->next != NULL) {  
        prev = temp;  
        temp = temp->next;  
    }  
    if (temp == head) {  
        head = NULL;  
    } else {  
        prev->next = NULL;  
    }  
    printf("Element deleted from the end: %d\n", temp->data);  
    free(temp);  
}  
void delete\_at\_position() {  
    int position;  
    printf("Enter the position to delete: ");  
    scanf("%d", &position);  
  
    if (head == NULL) {  
        printf("List is empty. Deletion not possible.\n");  
        return;  
    }  
    struct node \*temp, \*prev;  
    temp = head;  
    if (position == 0) {  
        head = head->next;  
        printf("Element at position %d deleted successfully.\n", position);  
        free(temp);  
        return;  
    }  
    for (int i = 0; temp != NULL && i < position; i++) {  
        prev = temp;  
        temp = temp->next;  
    }  
    if (temp == NULL) {  
        printf("Position %d is out of bounds.\n", position);  
        return;  
    }  
    prev->next = temp->next;  
    printf("Element at position %d deleted successfully.\n", position);  
    free(temp);  
}  
int main() {  
    int choice;  
    while (1) {  
        printf("\n 1. to insert at the beginning\n 2. to delete beginning\n 3. to delete at end\n 4. to delete at any position\n 5. to display\n 6. to exit\n");  
        printf("Enter your choice: ");  
        scanf("%d", &choice);  
        switch (choice) {  
            case 1:  
                insert\_begin();  
                break;  
            case 2:  
                delete\_begin();  
                break;  
            case 3:  
                delete\_end();  
                break;  
            case 4:  
                delete\_at\_position();  
                break;  
            case 5:  
                display();  
                break;  
            case 6:  
                exit(0);  
                break;  
            default:  
                printf("Enter the correct choice\n");  
                break;  
        }  
    }  
    return 0;  
}

output:-



Leet code :-

struct ListNode\* reverseBetween(struct ListNode\* start, int a, int b)

{

a-=1;

b-=1;

struct ListNode \*node1=NULL,\*node2=NULL,\*nodeb=NULL,\*nodea=NULL,\*ptr=start;

int c=0;

while(ptr!=NULL)

{

if(c==a-1)

nodeb=ptr;

else if(c==a)

node1=ptr;

else if(c==b)

node2=ptr;

else if(c==b+1)

{

nodea=ptr;

break;

}

c+=1;

ptr=ptr->next;

}

struct ListNode\*pre=nodea,\*temp;

ptr=start;

c=0;

while(ptr!=NULL)

{

if(c>=a && c<b)

{

temp=ptr->next;

ptr->next=pre;

pre=ptr;

ptr=temp;

}

else if(c==b)

{

ptr->next=pre;

if(a==0)

start=ptr;

else

nodeb->next=ptr;

break;

}

else

ptr=ptr->next;

c+=1;

}

return start;

}

Output:-

