LAB PROGRAM 5

WAP to Implement Singly Linked List with following operations

a) Create a linked list.

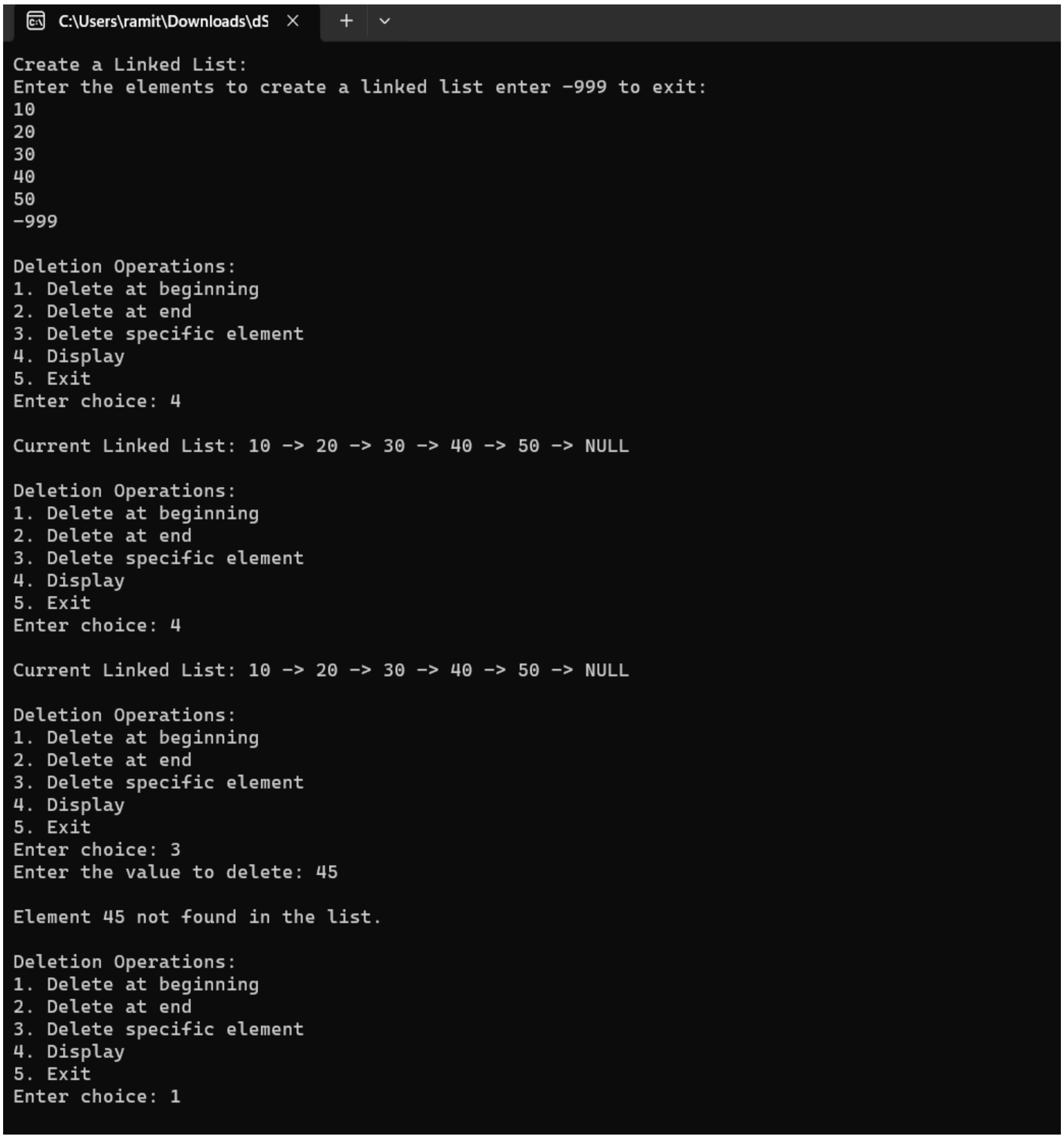
b) Deletion of first element, specified element and last element in the list

c) Display the contents of the linked list.

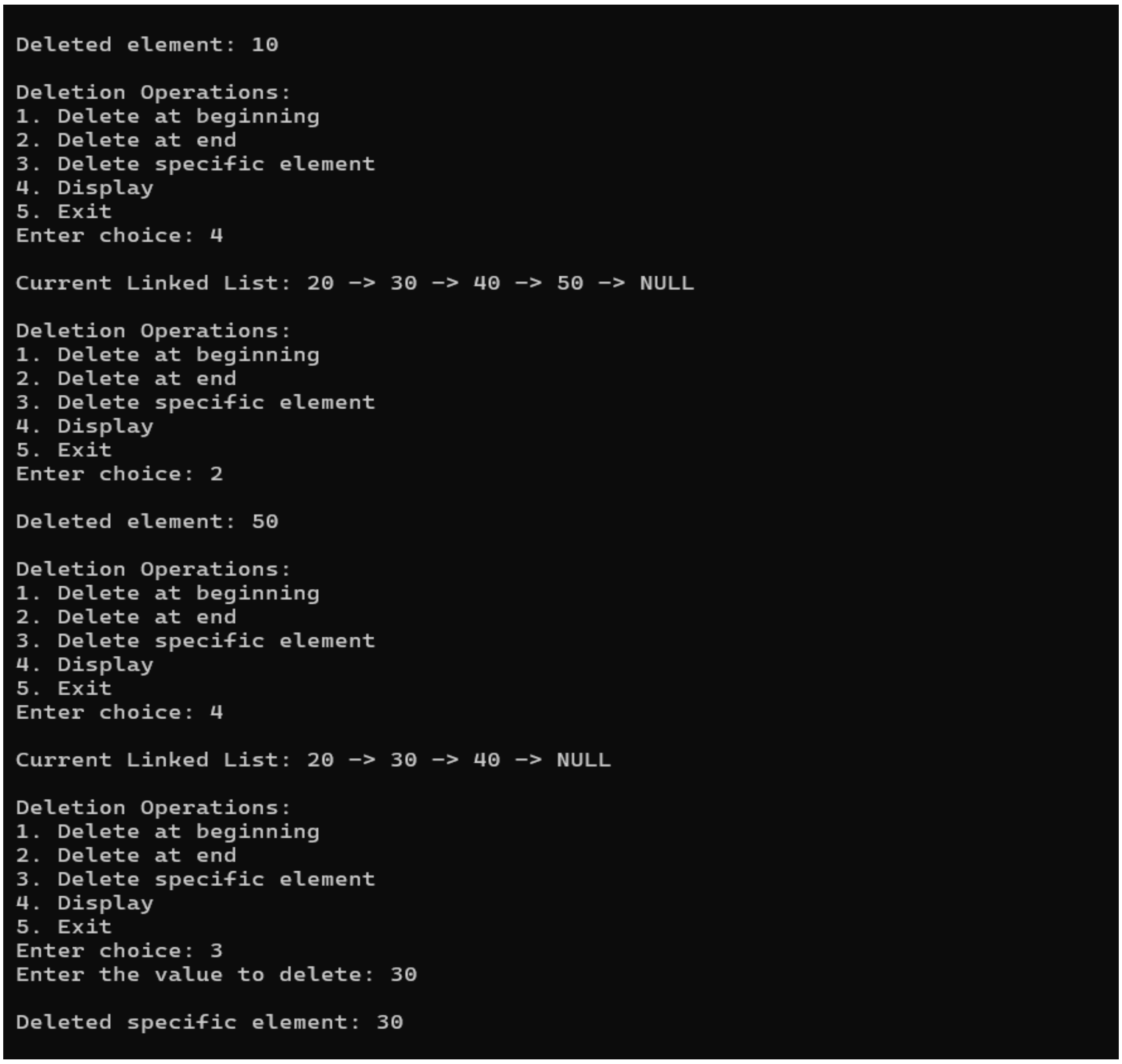
```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int val;
    struct node* next;
};

struct node * createsll(struct node *head)
{
    struct node *p , *last ;
    int ele;
    printf("Enter the elements to create a linked list enter -999 to
exit:\n");
    scanf("%d",&ele);
    while(ele!=-999)
    {
        p = (struct node *)malloc(sizeof(struct node));
        p->val = ele;
        if(head==NULL)
        {
            p->next=NULL;
            head=p;
            last=p;
        }
        else{
            last->next=p;
            p->next=NULL;
            last=p;
        }
        scanf("%d",&ele);
    }
    return head;
};

struct node * deleteAtBeginning(struct node *head) {
    if (head == NULL) {
        printf("\nList is already empty!\n");
    }
    struct node *temp = head;
    head = head->next;
    printf("\nDeleted element: %d\n", temp->val);
    free(temp);
    return head;
}


struct node * deleteAtEnd(struct node *head) {
    if (head == NULL) {
        printf("\nList is already empty!\n");

    }
    struct node* temp = head;
    struct node* prev = NULL;


    if (temp->next == NULL) {
        head = NULL;
        printf("\nDeleted element: %d\n", temp->val);
        free(temp);

    }


    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }

    prev->next = NULL;
    printf("\nDeleted element: %d\n", temp->val);
    free(temp);
    return head;
}


struct node * deleteSpecificElement(struct node *head) {
    if (head == NULL) {
        printf("\nList is already empty!\n");

    }

    int val;
    printf("Enter the value to delete: ");
    scanf("%d", &val);

    struct node* temp = head;
```

```c
    struct node* prev = NULL;


    if (temp != NULL && temp->val == val) {
        head = temp->next;
        printf("\nDeleted specific element: %d\n", val);
        free(temp);

    }


    while (temp != NULL && temp->val != val) {
        prev = temp;
        temp = temp->next;
    }


    if (temp == NULL) {
        printf("\nElement %d not found in the list.\n", val);

    }
    else
    {
        prev->next = temp->next;
        printf("\nDeleted specific element: %d\n", val);
        free(temp);

    }
    return head;



}

void display(struct node *head) {
    if (head == NULL) {
        printf("\nList is empty.\n");
        return;
    }
    struct node* temp = head;
    printf("\nCurrent Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->val);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct node *head = NULL;
    int choice;
    printf("Create a Linked List:\n");
    head=createsll(head);



    while (1) {

        printf("\nDeletion Operations:\n");
        printf("1. Delete at beginning\n");
        printf("2. Delete at end\n");
        printf("3. Delete specific element\n");
        printf("4. Display\n");
        printf("5. Exit\n");

        printf("Enter choice: ");
        scanf("%d", &choice);


        switch (choice) {
            case 1:
                head=deleteAtBeginning(head);
                break;
            case 2:
                head=deleteAtEnd(head);
                break;
            case 3:
                head=deleteSpecificElement(head);
                break;
            case 4:
                display(head);
                break;
            case 5:
                printf("Exiting program.\n");
                exit(0);
                break;
            default:
                printf("Invalid choice! Please enter a number between 1 and 
5.\n");
        }
    }

    return 0;
}
```

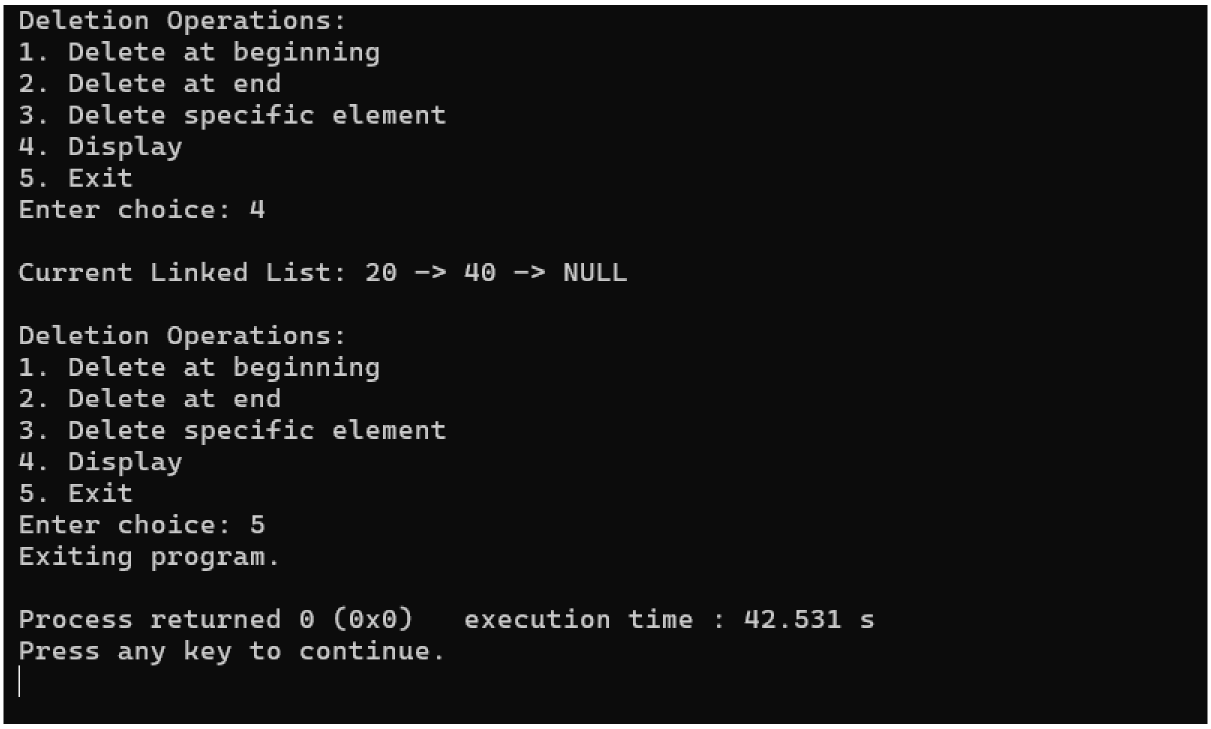
OUTPUT


C:\Users\ramit\Downloads\dS

```
Create a Linked List:
Enter the elements to create a linked list enter -999 to exit:
10
20
30
40
50
-999

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 4

Current Linked List: 10 -> 20 -> 30 -> 40 -> 50 -> NULL

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 4

Current Linked List: 10 -> 20 -> 30 -> 40 -> 50 -> NULL

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 3
Enter the value to delete: 45

Element 45 not found in the list.

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 1
```



```
Deleted element: 10

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 4

Current Linked List: 20 -> 30 -> 40 -> 50 -> NULL

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 2

Deleted element: 50

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 4

Current Linked List: 20 -> 30 -> 40 -> NULL

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 3
Enter the value to delete: 30

Deleted specific element: 30
```



```
Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 4

Current Linked List: 20 -> 40 -> NULL

Deletion Operations:
1. Delete at beginning
2. Delete at end
3. Delete specific element
4. Display
5. Exit
Enter choice: 5
Exiting program.

Process returned 0 (0x0)   execution time : 42.531 s
Press any key to continue.
```