5(a)_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>


// Define node structure

struct Node {

   int data;

   struct Node* next;

};


// Function to create a new node

struct Node* createNode(int data) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->data = data;

   newNode->next = NULL;

   return newNode;

}


// Function to insert at the end (used to create linked list)

struct Node* insertAtEnd(struct Node* head, int data) {

   struct Node* newNode = createNode(data);

   if (head == NULL) {

      head = newNode;

      return head;

   }

   struct Node* temp = head;

   while (temp->next != NULL)
```

```
        temp = temp->next;
    temp->next = newNode;
    return head;
}

// Function to delete first element
struct Node* deleteFirst(struct Node* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return head;
    }
    struct Node* temp = head;
    head = head->next;
    free(temp);
    return head;
}

// Function to delete last element
struct Node* deleteLast(struct Node* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return head;
    }
    if (head->next == NULL) {
        free(head);
        return NULL;
    }
    struct Node* temp = head;
```

```
    while (temp->next->next != NULL)
        temp = temp->next;
    free(temp->next);
    temp->next = NULL;
    return head;
}

// Function to delete a specified element by value
struct Node* deleteElement(struct Node* head, int key) {
    if (head == NULL) {
        printf("List is empty.\n");
        return head;
    }
    if (head->data == key) {
        struct Node* temp = head;
        head = head->next;
        free(temp);
        return head;
    }
    struct Node* temp = head;
    while (temp->next != NULL && temp->next->data != key)
        temp = temp->next;
    if (temp->next == NULL) {
        printf("Element %d not found in the list.\n", key);
        return head;
    }
    struct Node* delNode = temp->next;
    temp->next = delNode->next;
```

```
    free(delNode);
    return head;
}

// Function to display the linked list
void displayList(struct Node* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head;
    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;
    int choice, data;

    while (1) {
        printf("\n1. Create/Add node at end\n2. Delete first node\n3. Delete last node\n4. Delete specified element\n5. Display\n6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
    switch (choice) {
        case 1:
            printf("Enter data: ");
            scanf("%d", &data);
            head = insertAtEnd(head, data);
            break;
        case 2:
            head = deleteFirst(head);
            break;
        case 3:
            head = deleteLast(head);
            break;
        case 4:
            printf("Enter element to delete: ");
            scanf("%d", &data);
            head = deleteElement(head, data);
            break;
        case 5:
            displayList(head);
            break;
        case 6:
            exit(0);
        default:
            printf("Invalid choice.\n");
    }
}
```

return 0;

```
1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 1
Enter data: 10

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 1
Enter data: 20

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 1
Enter data: 30

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 5
Linked List: 10 -> 20 -> 30 -> NULL
```

}

```
1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 2

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 5
Linked List: 20 -> 30 -> NULL

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 4
Enter element to delete: 30

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 5
Linked List: 20 -> NULL
```
}

```
1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 3

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 3
List is empty.

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 5
List is empty.

1. Create/Add node at end
2. Delete first node
3. Delete last node
4. Delete specified element
5. Display
6. Exit
Enter your choice: 3
List is empty.
```