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

struct node {
    int data;
    struct node *next;
};
struct node *head = NULL;

void createList(int n) {
    struct node *newNode, *temp = NULL;
    int data;
    printf("Enter %d elements:\n", n);
    for (int i = 1; i <= n; i++) {
        newNode = malloc(sizeof(struct node));
        if (!newNode) {
            printf("Memory allocation failed.\n");
            return;
        }
        printf("Element %d: ", i);
        scanf("%d", &data);
        newNode->data = data;
        newNode->next = NULL;
        if (head == NULL)
            head = temp = newNode;
        else {
            temp->next = newNode;
            temp = newNode;
        }
    }
    printf("List created successfully.\n");
}

void deleteAtbeginning(){
    if(head==NULL){
        printf("List is empty\n");return;}
    struct node* temp = head;
    head = head->next;
    free(temp);
}

void deleteAtEnd(){
    if (head == NULL){printf("List is empty\n");return;}
    if(head->next == NULL){
        free(head);
        head=NULL;return;
    }
    struct node *temp;
    struct node *tail;
    temp=head;
    while(temp->next != NULL){
        tail=temp;
        temp =temp->next;
```

```c
    }
    tail->next = NULL;
    free(temp);
}
void deleteAtanyPos(int pos){
    if(head == NULL){printf("List is empty\n");return;}
    if(pos == 1){deleteAtbeginning();return;}
    struct node *temp,*tail;
    temp=head;int i=1;
    while(i<pos-1 && temp!=NULL){
        tail = temp;
        temp = temp->next;i++;
    }
    if(temp == NULL | temp->next == NULL){
        printf("Invalid position\n");return;
    }
    tail->next=temp->next;
    free(temp);
}
void displayList() {
    struct node *temp = head;
    if (!head) {
        printf("List is empty.\n");
        return;
    }
    printf("Current List: ");
    while (temp) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice, n, data, pos;

    while (1) {
        printf("\n---- Singly Linked List operation ----\n");
        printf("1. Create List\n");
        printf("2. Delete at Beginning\n");
        printf("3. Delete at Position\n");
        printf("4. Delete at End\n");
        printf("5. Display List\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```c
int main() {
    int choice, n, data, pos;

    while (1) {
        printf("\n---- Singly Linked List operation ----\n");
        printf("1. Create List\n");
        printf("2. Delete at Beginning\n");
        printf("3. Delete at Position\n");
        printf("4. Delete at End\n");
        printf("5. Display List\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
             printf("Enter number of nodes: ");
             scanf("%d", &n);
             createList(n);break;
        case 2:
            deleteAtbeginning();break;
        case 3:
             printf("Enter position of node: ");
             scanf("%d",&pos);
             deleteAtanyPos(pos);break;
        case 4:
            deleteAtEnd();break;
        case 5:
                displayList();break;
        case 6:
                printf("Exiting...\n");
                return 0;
        default:
                printf("Invalid choice! Try again.\n");break;
        }

    }
}
```

```
---- Singly Linked List operation ----
1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 1
Enter number of nodes: 4
Enter 4 elements:
Element 1: 10
Element 2: 20
Element 3: 30
Element 4: 40
List created successfully.

---- Singly Linked List operation ----
1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 2

---- Singly Linked List operation ----
1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 5
Current List: 20 -> 30 -> 40 -> NULL

---- Singly Linked List operation ----
1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 3
Enter position of node: 3

---- Singly Linked List operation ----
1. Create List
2. Delete at Beginning
3. Delete at Position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 5
Current List: 20 -> 40 -> NULL
```

Lab 5

(a) To implement deletion of Node in a Linked list a) creation of LL b) display the contents of the Linked list.

pseudocode:-

```
Function Delete_At_Begin()
    if head == NULL
        print "List is Empty"
        return
    temp = Head;
    Head = head.next;
    delete temp;
END Function

Function DELETE_AT_END()
    if head == NULL
        print "List is Empty"
        return
    if head.next = NULL
        DeleteAt_Begin()
        return
    temp = head
    while temp.next != NULL
        tail = temp
        temp = temp.next
    End while
    tail = temp
    tail.next = temp.next NULL
    delete temp
End function.

Function. Delete_At_pos(pos)
    if Head == NULL
        print "List is Empty
    if pos == 1
        DELETE_At_Begin()
        return
    temp = Head
    i = 1
    while i < pos-1 and temp != NULL
        temp = temp.next
        i++
    if temp == NULL or temp.next == NULL
        print "invalid position"
        return
```

```
nodeToDelete = temp->next
tempNext = nodeToDelete->next
delete nodeToDelete
```

Code:-

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
};
struct node* head = NULL;
void createList(int n) {
    struct node *newNode, *temp = NULL;
    int data;
    printf("Enter %d element:\n", n);
    for (int i=1; i<=n; i++) {
        newNode = malloc(sizeof(struct node));
        if (!newNode) {
            printf("memory allocation failed\n"); return;
        }
        printf("Element %d: ", i);
        scanf("%d", &data);
        newNode->data = data;
        newNode->next = NULL;
        if (head == NULL) { head = temp = newNode; }
        else {
            temp->next = newNode;
            temp = newNode;
        }
    }
```

```
void deleteAtEnd() {
    if (head == NULL) { printf("List is Empty\n"); return; }
    if (head->next == NULL) {
        free(head);
        head = NULL;
        return;
    }
    struct node *temp;
    struct node *tail;
    temp = head;
    while (temp->next != NULL) {
        tail = temp;
        temp = temp->next;
    }
    tail->next = NULL;
    free(temp);
}

void deleteAtBeginning() {
    if (head == NULL) {
        printf("List is Empty\n"); return;
    }
    struct node *temp = head;
    head = head->next;
    free(temp);
}

void displayAtanyPos(int pos) {
    if (head == NULL) { printf("List is empty\n"); return; }
    if (pos == 1) { deleteAtBeginning(); return; }
    struct node *temp, *tail;
    temp = head; int i = 1;
    while (i<pos-1 && temp!=NULL) {
        tail = temp;
        temp = temp->next; i++;
    }
    if (temp == NULL || temp->next == NULL) {
        printf("Invalid position\n"); return;
    }
    tail->next = temp->next;
    free(temp);
}
```

```
void displayList() {
    struct node *temp = head;
    if (!head) {
        printf("List is empty\n");
        return;
    }
    printf("Current List: ");
    while (temp) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("\n---- Singly Linked List operations ----");
        printf("1. Create List\n");
        printf("2. Delete at Beginning\n");
        printf("3. Delete at position\n");
        printf("4. Delete at End\n");
        printf("5. Display List\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n); break;
            case 2:
                deleteAtBeginning(); break;
            case 3:
                printf("Enter position of node: ");
                scanf("%d", &pos);
                deleteAtPos(pos); break;
            case 4:
                deleteAtEnd(); break;
            case 5:
                displayList(); break;
            case 6:
                printf("Exiting....\n");
                return 0;
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
}
```

o/p:-

```
---- Singly Linked List operations ----
1. CreateList
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 1
Enter number of nodes: 4
Enter 4 Elements:
Element 1: 10
Element 2: 20
Element 3: 30
Element 4: 40
List created Sucessfully.

---- Singly Linked List operations ----
1. CreateList
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 2

---- Singly Linked List operations ----
1. CreateList
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit
Enter your choice: 5
Current List: 20 -> 30 -> 40 -> NULL
```

---- Singly Linked List operation ----
1. Create List
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display List
6. Exit
Enter your choice : 3
Enter position of node : 3

---- Singly Linked List Operation ----
1. CreateList
2. Delete at Beginning
3. Delete at position
4. Delete at End
5. Display ~~List~~
6. Exit
Enter your choice : 5
Current List : 20 → 30 → NULL