ACPCE
Where knowledge is second nature

Jawahar Education Society's
A. C. Patil College of Engineering, Kharghar
Navi Mumbai 410210

Student Name: Manish gangole

PRN No.: 221111028

Course Name: C.S.E. (IoT CS BC)

Course code: CSL301

Year: S.E.

Semester: 3

Roll No.:

Experiment Evaluation Sheet

Experiment No.: 2

Experiment Name:
Program to create a singly linked list and perform the operations like insertion, deletion, display & count the nodes.

Sr No.	Evaluation Criteria	Marks (Out of 9)	Performance Date	Correction Date and Signature of Instructor
1	Experiment Performance			
2	Journal Performance			
3	Punctuality			
Total				

Code :

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

// Define the structure for a node in the linked list
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));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a node at the beginning of the linked list
struct Node* insertAtStart(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = head;
    return newNode;
}

// Function to insert a node at the end of the linked list
struct Node* insertAtEnd(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    if (head == NULL) {
        return newNode;
    }
    struct Node* current = head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
    return head;
}

// Function to insert a node after a specific data value in the linked list
struct Node* insertAfterValue(struct Node* head, int data, int newValue) {
    struct Node* newNode = createNode(data);
    struct Node* current = head;

    while (current != NULL) {
        if (current->data == newValue) {
            newNode->next = current->next;
            current->next = newNode;
            return head;
        }
        current = current->next;
```

Code :

```
    }

    printf("Value %d not found in the list. Node not inserted.\n", newValue);
    free(newNode);
    return head;
}


// Function to delete a node with a given value from the linked list
struct Node* deleteNode(struct Node* head, int key) {
    struct Node* current = head;
    struct Node* prev = NULL;
    while (current != NULL && current->data != key) {
        prev = current;
        current = current->next;
    }
    if (current == NULL) {
        printf("Element not found in the list!\n");
        return head;
    }
    if (prev == NULL) {
        // If the first node is the key
        struct Node* temp = head;
        head = head->next;
        free(temp);
    } else {
        prev->next = current->next;
        free(current);
    }
    return head;
}

// Function to display the linked list
void displayList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

// Function to count the number of nodes in the linked list
int countNodes(struct Node* head) {
    int count = 0;
    struct Node* current = head;
    while (current != NULL) {
        count++;
        current = current->next;
    }
    return count;
}
```

Code :

```
int main() {
    struct Node* head = NULL;
    int choice, data, position;
    while (1) {
        printf("\nLinked List Operations:\n");
        printf("1. Insert at the start\n");
        printf("2. Insert at the end\n");
        printf("3. Insert in the middle\n");
        printf("4. Delete an element\n");
        printf("5. Display the linked list\n");
        printf("6. Count the number of nodes\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the data to insert at the start: ");
                scanf("%d", &data);
                head = insertAtStart(head, data);
                break;
            case 2:
                printf("Enter the data to insert at the end: ");
                scanf("%d", &data);
                head = insertAtEnd(head, data);
                break;
            case 3:
                printf("Enter the data to insert: ");
                scanf("%d", &data);
                printf("Enter the value after which you want to insert: ");
                scanf("%d", &position);
                head = insertAfterValue(head, data, position);
                break;
            case 4:
                printf("Enter the element to delete: ");
                scanf("%d", &data);
                head = deleteNode(head, data);
                break;
            case 5:
                printf("Linked List: ");
                displayList(head);
                break;
            case 6:
                printf("Number of nodes in the linked list: %d\n", countNodes(head));
                break;
            case 7:
                printf("Exiting program.\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

Output :

```
Linked List Operations:
1. Insert at the start
2. Insert at the end
3. Insert in the middle
4. Delete an element
5. Display the linked list
6. Count the number of nodes
7. Exit
Enter your choice: 1
Enter the data to insert at the start: 25

Linked List Operations:
1. Insert at the start
2. Insert at the end
3. Insert in the middle
4. Delete an element
5. Display the linked list
6. Count the number of nodes
7. Exit
Enter your choice: 2
Enter the data to insert at the end: 34

Linked List Operations:
1. Insert at the start
2. Insert at the end
3. Insert in the middle
4. Delete an element
5. Display the linked list
6. Count the number of nodes
7. Exit
Enter your choice: 3
Enter the data to insert: 69
Enter the value after which you want to insert: 25
```

```
Enter your choice: 4
Enter the element to delete: 34

Linked List Operations:
1. Insert at the start
2. Insert at the end
3. Insert in the middle
4. Delete an element
5. Display the linked list
6. Count the number of nodes
7. Exit
Enter your choice: 5
Linked List: 25 -> 69 -> NULL

Linked List Operations:
1. Insert at the start
2. Insert at the end
3. Insert in the middle
4. Delete an element
5. Display the linked list
6. Count the number of nodes
7. Exit
Enter your choice: 6
Number of nodes in the linked list: 2

Linked List Operations:
1. Insert at the start
2. Insert at the end
3. Insert in the middle
4. Delete an element
5. Display the linked list
6. Count the number of nodes
7. Exit
Enter your choice: 7
Exiting program.
```

Conclusion :

Through this experiment we have learnt about how to implement a linked list using the C language.
Various operations like insertion, deletion, display & count are applied on the linked list.
This experiment helps us in using arrays as a data structure for further reference.