SEM 3\Exp1\SLL_implementation.c

```c
// program to implement singly linked list

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

// Core structure of Node that forms Linked List
struct node
{
    int data;
    struct node *next;
};

// End of the linked list should not point to anything(NULL)
struct node *head = NULL;

// Function to insert a node at the beginning of the list
void insertFirst(int data)
{
    struct node *new_node = (struct node *)malloc(sizeof(struct node));

    new_node->data = data;
    new_node->next = head;

    head = new_node;
}

// Function to insert a node at the end of the list
void insertEnd(int data)
{
    struct node *new_node = (struct node *)malloc(sizeof(struct node));

    new_node->data = data;
    new_node->next = NULL;

    if (head == NULL)
    {
        head = new_node;
        return;
    }

    struct node *temp = head;

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

    temp->next = new_node;
}

void insertPos(int data, int pos)
```

```c
{
    struct node *new_node = (struct node *)malloc(sizeof(struct node));

    new_node->data = data;

    int curr_pos = 0;
    struct node *temp = head;

    while (temp->next != NULL && curr_pos < pos - 1)
    {
        temp = temp->next;
        curr_pos++;
    }

    new_node->next = temp->next;
    temp->next = new_node;
}

void deleteFirst()
{
    if (head == NULL)
    {
        printf("List is empty");
        return;
    }

    struct node *temp = head;
    head = head->next;
    free(temp);
}

void deleteEnd()
{
    if (head == NULL)
    {
        printf("List is empty");
        return;
    }

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

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

    prev->next = NULL;
    free(temp);
}

void deletePos(int pos)
{
```

```c
    if (head == NULL)
    {
        printf("List is empty");
        return;
    }

    struct node *temp = head;
    struct node *prev = NULL;
    int curr_pos = 0;

    while (temp->next != NULL && curr_pos < pos - 1)
    {
        prev = temp;
        temp = temp->next;
        curr_pos++;
    }

    prev->next = temp->next;
    free(temp);
}

void display()
{
    struct node *temp = head;

    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main()
{
    printf("Linked List creation and Manipulation\n");
    printf("Enter from the following options:\n");
    printf("1. Insert at the beginning of the list\n");
    printf("2. Insert at the end of the list\n");
    printf("3. Insert at a specific position in the list\n");
    printf("4. Delete from the beginning of the list\n");
    printf("5. Delete from the end of the list\n");
    printf("6. Delete from a specific position in the list\n");
    printf("7. Display the list\n");
    printf("8. Exit\n");

    int choice;
    int data;
    int pos;

    while (1)
    {
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```c

        switch (choice)
        {
        case 1:
            printf("Enter the data to be inserted: ");
            scanf("%d", &data);
            insertFirst(data);
            break;
        case 2:
            printf("Enter the data to be inserted: ");
            scanf("%d", &data);
            insertEnd(data);
            break;
        case 3:
            printf("Enter the data to be inserted: ");
            scanf("%d", &data);
            printf("Enter the position to insert the data: ");
            scanf("%d", &pos);
            insertPos(data, pos);
            break;
        case 4:
            deleteFirst();
            break;
        case 5:
            deleteEnd();
            break;
        case 6:
            printf("Enter the position to delete the data: ");
            scanf("%d", &pos);
            deletePos(pos);
            break;
        case 7:
            display();
            break;
        case 8:
            exit(0);
        default:
            printf("Invalid choice");
            break;
        }
    }

    return 0;
}
```