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

// Structure to represent Student Data
struct Student {
    char USN[15];
    char Name[30];
    char Programme[20];
    int Sem;
    long long int PhNo;
    struct Student* next;
};

// Function to create a new node with given data
struct Student* createNode() {
    struct Student* newNode = (struct
Student*)malloc(sizeof(struct Student));

    printf("Enter USN: ");
    scanf("%s", newNode->USN);
    printf("Enter Name: ");
    scanf("%s", newNode->Name);
    printf("Enter Programme: ");
    scanf("%s", newNode->Programme);
    printf("Enter Semester: ");
    scanf("%d", &newNode->Sem);
    printf("Enter Phone Number: ");
    scanf("%lld", &newNode->PhNo);

    newNode->next = NULL;
    return newNode;
}

// Function to insert at the front of the list
void insertFront(struct Student** head) {
    struct Student* newNode = createNode();
    newNode->next = *head;
    *head = newNode;
    printf("Node inserted at the front successfully.\n");
}

// Function to display the status of the linked list and count
nodes
void displayStatus(struct Student* head) {
    if (head == NULL) {
```

```c
        printf("List is empty.\n");
        return;
    }

    printf("Student Data in the Linked List:\n");
    struct Student* current = head;
    int count = 0;

    while (current != NULL) {
        printf("USN: %s, Name: %s, Programme: %s, Sem: %d, 
Phone: %lld\n",
               current->USN, current->Name, current->Programme, 
current->Sem, current->PhNo);
        current = current->next;
        count++;
    }

    printf("Total number of nodes: %d\n", count);
}

// Function to insert at the end of the list
void insertEnd(struct Student** head) {
    struct Student* newNode = createNode();

    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Student* current = *head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newNode;
    }

    printf("Node inserted at the end successfully.\n");
}

// Function to delete from the front of the list
void deleteFront(struct Student** head) {
    if (*head == NULL) {
        printf("List is empty. Nothing to delete.\n");
        return;
    }

    struct Student* temp = *head;
    *head = (*head)->next;
    free(temp);
    printf("Node deleted from the front successfully.\n");
}

// Function to delete from the end of the list
void deleteEnd(struct Student** head) {
    if (*head == NULL) {
        printf("List is empty. Nothing to delete.\n");
        return;
    }

    if ((*head)->next == NULL) {
        free(*head);
        *head = NULL;
        printf("Node deleted from the end successfully.\n");
        return;
    }

    struct Student* current = *head;
    struct Student* prev = NULL;

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

    free(current);
    prev->next = NULL;
    printf("Node deleted from the end successfully.\n");
}

// Function to free the entire linked list
void freeList(struct Student** head) {
    struct Student* current = *head;
    struct Student* next;

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

    *head = NULL;
}
```

```
int main() {
    struct Student* head = NULL;
    int choice;

    do {
        printf("\nMenu:\n");
        printf("1. Insert at Front\n");
        printf("2. Display Status\n");
        printf("3. Insert at End\n");
        printf("4. Delete from Front (Demonstration of stack)
\n");
        printf("5. Delete from End\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertFront(&head);
                break;
            case 2:
                displayStatus(head);
                break;
            case 3:
                insertEnd(&head);
                break;
            case 4:
                deleteFront(&head);
                break;
            case 5:
                deleteEnd(&head);
                break;
            case 6:
                freeList(&head);
                printf("Exiting the program.\n");
                break;
            default:
                printf("Invalid choice. Please enter a valid
option.\n");
        }
    } while (choice != 6);

    return 0;
}
```