Program 07 : Singly Linked List of Student Data

Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo

a. Create a SLL of N Students Data by using front insertion.
b. Display the status of SLL and count the number of nodes in it
c. Perform Insertion / Deletion at End of SLL
d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
e. Exit

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct Student {
   char USN[15];
   char Name[50];
   char Programme[50];
   int Sem;
   char PhNo[15];
   struct Student *next;
} Student;
// Function to create a new student node
Student* createStudent() {
   Student *newStudent = (Student*)malloc(sizeof(Student));
   printf("Enter USN: ");
   scanf("%s", newStudent->USN);
   printf("Enter Name: ");
   scanf("%s", newStudent->Name); // Use fgets in real-world applications
   printf("Enter Programme: ");
   scanf("%s", newStudent->Programme);
   printf("Enter Semester: ");
   scanf("%d", &newStudent->Sem);
   printf("Enter Phone Number: ");
   scanf("%s", newStudent->PhNo);
   newStudent->next = NULL;
   return newStudent;
}
```

```
// Function to insert a student at the front of the list
void insertFront(Student **head) {
   Student *newStudent = createStudent();
   newStudent->next = *head;
   *head = newStudent;
   printf("Student added at front.\n");
}
// Function to insert a student at the end of the list
void insertEnd(Student **head) {
   Student *newStudent = createStudent();
   if (*head == NULL) {
      *head = newStudent;
   } else {
      Student *temp = *head;
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newStudent;
   }
   printf("Student added at end.\n");
}
// Function to delete a student from the front of the list
void deleteFront(Student **head) {
   if (*head == NULL) {
      printf("List is empty, nothing to delete.\n");
      return;
   }
   Student *temp = *head;
   *head = (*head)->next;
   free(temp);
   printf("Student deleted from front.\n");
}
// Function to delete a student from the end of the list
void deleteEnd(Student **head) {
   if (*head == NULL) {
      printf("List is empty, nothing to delete.\n");
      return;
   }
   if ((*head)->next == NULL) {
      free(*head);
```

```
        *head = NULL;
    } else {
        Student *temp = *head;
        while (temp->next->next != NULL) {
            temp = temp->next;
        }
        free(temp->next);
        temp->next = NULL;
    }
    printf("Student deleted from end.\n");
}
// Function to display the list and count the number of nodes
void displayAndCount(Student *head) {
    int count = 0;
    printf("\nStudent List:\n");
    while (head != NULL) {
        printf("USN: %s, Name: %s, Programme: %s, Semester: %d, Phone: %s\n",
               head->USN, head->Name, head->Programme, head->Sem, head->PhNo);
        head = head->next;
        count++;
    }
    printf("Total Students: %d\n", count);
}
int main() {
    Student *head = NULL; // Initialize the head of the list
    int choice;
    while (1) {
        printf("\n--- Singly Linked List Operations ---\n");
        printf("1. Insert Student at Front\n");
        printf("2. Display SLL and Count Nodes\n");
        printf("3. Insert at End\n");
        printf("4. Delete at Front\n");
        printf("5. Delete at End\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
        switch (choice) {
            case 1:
                insertFront(&head);
                break;
            case 2:
                displayAndCount(head);
                break;
            case 3:
                insertEnd(&head);
                break;
            case 4:
                deleteFront(&head);
                break;
            case 5:
                deleteEnd(&head);
                break;
            case 6:
                printf("Exiting...\n");
                // Free the list before exiting
                while (head != NULL) {
                    Student *temp = head;
                    head = head->next;
                    free(temp);
                }
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```