Program 08 : Doubly Linked List of Employee Data

Develop a menu driven Program in C for the following operations on Doubly Linked List(DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo
a. Create a DLL of N Employees Data by using end insertion.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL
e. Demonstrate how this DLL can be used as Double Ended Queue.
f. Exit

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Employee Structure

typedef struct Employee {

    char ssn[20];

    char name[100];

    char dept[50];

    char designation[50];

    double sal;

    char phNo[15];

    struct Employee *prev, *next;

} Employee;

// Function Declarations

Employee* createEmployee();

Employee* insertAtEnd(Employee* head);

void display(Employee* head);
```

```
int countNodes(Employee* head);
Employee* deleteAtEnd(Employee* head);
Employee* insertAtFront(Employee* head);
Employee* deleteAtFront(Employee* head);
// Main Function
int main() {
    Employee *head = NULL;
    int choice, count;
    while(1) {
        printf("\nMenu:\n");
        printf("1. Insert Employee at End\n");
        printf("2. Display Employees\n");
        printf("3. Count Employees\n");
        printf("4. Delete Employee at End\n");
        printf("5. Insert Employee at Front\n");
        printf("6. Delete Employee at Front\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1: head = insertAtEnd(head); break;
            case 2: display(head); break;
            case 3:
```

```
            count = countNodes(head);
            printf("Total number of employees: %d\n", count);
            break;
        case 4: head = deleteAtEnd(head); break;
        case 5: head = insertAtFront(head); break;
        case 6: head = deleteAtFront(head); break;
        case 7: exit(0);
        default: printf("Invalid choice!\n");
      }
   }
   return 0;
}
// Function Definitions
Employee* createEmployee() {
   Employee *newNode = (Employee*)malloc(sizeof(Employee));
   printf("Enter SSN: ");
   scanf("%s", newNode->ssn);
   printf("Enter Name: ");
   scanf("%s", newNode->name); // Use fgets for multi-word names
   printf("Enter Department: ");
   scanf("%s", newNode->dept);
   printf("Enter Designation: ");
   scanf("%s", newNode->designation);
   printf("Enter Salary: ");
```

```
        fflush(stdin);
    scanf("%lf", &newNode->sal);
    printf("Enter Phone Number: ");
    scanf("%s", newNode->phNo);

    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
Employee* insertAtEnd(Employee* head) {
    Employee *newNode = createEmployee();
    if (head == NULL) {
        return newNode;
    } else {
        Employee *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
        return head;
    }
}
```

```
void display(Employee* head) {
    Employee *temp = head;
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }
    while (temp != NULL) {
        printf("SSN: %s, Name: %s, Dept: %s, Designation: %s, Salary: %.2f, Phone No: %s\n",
               temp->ssn, temp->name, temp->dept, temp->designation, temp->sal, temp->phNo);
        temp = temp->next;
    }
}
int countNodes(Employee* head) {
    int count = 0;
    Employee *temp = head;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }
    return count;
}
Employee* deleteAtEnd(Employee* head) {
    if (head == NULL) {
```

```
        printf("List is already empty!\n");
        return NULL;
    } else if (head->next == NULL) {
        free(head);
        return NULL;
    } else {
        Employee *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->prev->next = NULL;
        free(temp);
        return head;
    }
}

Employee* insertAtFront(Employee* head) {
    Employee *newNode = createEmployee();
    if (head == NULL) {
        return newNode;
    } else {
        newNode->next = head;
        head->prev = newNode;
        return newNode;
```

```
    }
}
Employee* deleteAtFront(Employee* head) {
    if (head == NULL) {
        printf("List is already empty!\n");
        return NULL;
    } else {
        Employee *temp = head;
        head = head->next;
        if (head != NULL) head->prev = NULL;
        free(temp);
        return head;
    }
}
```