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

// Structure to represent Employee Data
struct Employee {
    char SSN[15];
    char Name[30];
    char Dept[20];
    char Designation[30];
    float Sal;
    long long int PhNo;
    struct Employee* next;
    struct Employee* prev;
};

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

    printf("Enter SSN: ");
    scanf("%s", newNode->SSN);
    printf("Enter Name: ");
    scanf("%s", newNode->Name);
    printf("Enter Dept: ");
    scanf("%s", newNode->Dept);
    printf("Enter Designation: ");
    scanf("%s", newNode->Designation);
    printf("Enter Salary: ");
    scanf("%f", &newNode->Sal);
    printf("Enter Phone Number: ");
    scanf("%lld", &newNode->PhNo);

    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

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

    if (*head == NULL) {
        *head = *tail = newNode;
    } else {
        newNode->prev = *tail;
        (*tail)->next = newNode;
        *tail = newNode;
    }

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

// Function to display the status of the doubly linked list and
count nodes
void displayStatus(struct Employee* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

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

    while (current != NULL) {
        printf("SSN: %s, Name: %s, Dept: %s, Designation: %s,
Salary: %.2f, Phone: %lld\n",
               current->SSN, current->Name, current->Dept,
current->Designation, current->Sal, current->PhNo);
        current = current->next;
        count++;
    }

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

// Function to insert at the front of the list
void insertFront(struct Employee** head, struct Employee**
tail) {
    struct Employee* newNode = createNode();

    if (*head == NULL) {
        *head = *tail = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
```

```c
    printf("Node inserted at the front successfully.\n");
}

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

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

    struct Employee* temp = *tail;
    *tail = (*tail)->prev;
    (*tail)->next = NULL;
    free(temp);

    printf("Node deleted from the end successfully.\n");
}

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

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

    struct Employee* temp = *head;
    *head = (*head)->next;
    (*head)->prev = NULL;
    free(temp);

    printf("Node deleted from the front successfully.\n");
}

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

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

    *head = *tail = NULL;
}

int main() {
    struct Employee* head = NULL;
    struct Employee* tail = NULL;
    int choice;

    do {
        printf("\nMenu:\n");
        printf("1. Insert at End\n");
        printf("2. Display Status\n");
        printf("3. Insert at Front\n");
        printf("4. Delete from End\n");
        printf("5. Delete from Front\n");
        printf("6. Demonstrate as Double Ended Queue\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertEnd(&head, &tail);
                break;
            case 2:
                displayStatus(head);
                break;
            case 3:
                insertFront(&head, &tail);
                break;
            case 4:
                deleteEnd(&head, &tail);
                break;
            case 5:
                deleteFront(&head, &tail);
                break;
            case 6:
                // Demonstrate as Double Ended Queue
                insertFront(&head, &tail); // Enqueue at front
                deleteEnd(&head, &tail);   // Dequeue from end
                break;
            case 7:
                freeList(&head, &tail);
                printf("Exiting the program.\n");
                break;
            default:
                printf("Invalid choice. Please enter a valid 
option.\n");
        }
    } while (choice != 7);

    return 0;
}
```