SEM 3\Exp2\DLL_implementation.c

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

// Node structure for the doubly linked list
struct Node
{
    int data;
    struct Node *prev;
    struct Node *next;
};

// Insert at the end of the doubly linked list
void insert(struct Node **head_ref, int new_data)
{
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    struct Node *last = *head_ref;
    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL)
    {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = new_node;
    new_node->prev = last;
}

// Display the doubly linked list
void display(struct Node *node)
{
    struct Node *last;
    printf("Traversal in forward direction:\n");
    while (node != NULL)
    {
        printf("%d ", node->data);
        last = node;
        node = node->next;
    }

    printf("\nTraversal in reverse direction:\n");
    while (last != NULL)
    {
        printf("%d ", last->data);
        last = last->prev;
    }
    printf("\n");
}

// Delete a node from the doubly linked list
void deleteNode(struct Node **head_ref, int key)
{
    struct Node *temp = *head_ref;

    if (*head_ref == NULL)
        return;

    while (temp != NULL && temp->data != key)
        temp = temp->next;

    if (temp == NULL)
        return;

    if (*head_ref == temp)
        *head_ref = temp->next;

    if (temp->next != NULL)
        temp->next->prev = temp->prev;

    if (temp->prev != NULL)
        temp->prev->next = temp->next;

    free(temp);
}

// Search for a key in the doubly linked list
void search(struct Node *head, int key)
{
    struct Node *temp = head;
    int pos = 0;
    while (temp != NULL)
    {
        if (temp->data == key)
        {
            printf("Element %d found at position %d\n", key, pos);
            return;
        }
        temp = temp->next;
        pos++;
    }
    printf("Element %d not found in the list\n", key);
}

// Count the number of nodes in the doubly linked list
int count(struct Node *head)
{
    int count = 0;
    struct Node *temp = head;
    while (temp != NULL)
    {
        count++;
        temp = temp->next;
    }
    return count;
}

int main()
{
    struct Node *head = NULL;
    int choice, value, key;

    printf("\nDoubly Linked List Operations:\n");
    printf("1. Insert\n");
    printf("2. Display\n");
    printf("3. Delete\n");
    printf("4. Search\n");
    printf("5. Count\n");
    printf("6. Exit\n");

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

        switch (choice)
        {
        case 1:
            printf("Enter the value to insert: ");
            scanf("%d", &value);
            insert(&head, value);
            printf("\n");
            break;
        case 2:
            display(head);
            printf("\n");
            break;
        case 3:
            printf("Enter the value to delete: ");
            scanf("%d", &key);
            deleteNode(&head, key);
            printf("\n");
            break;
        case 4:
            printf("Enter the value to search: ");
            scanf("%d", &key);
            search(head, key);
            printf("\n");
            break;
        case 5:
            printf("The number of nodes in the list: %d\n", count(head));
            printf("\n");
            break;
        case 6:
            exit(0);
        default:
            printf("Invalid choice!\n");
        }
    }

    return 0;
}

```