SEM 3\Exp3\CSLL.c

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

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

// Insert a new node at the end of the circular linked list
void insert(struct Node **head_ref, int new_data)
{
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    struct Node *temp = *head_ref;
    new_node->data = new_data;
    new_node->next = *head_ref;

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

    while (temp->next != *head_ref)
        temp = temp->next;

    temp->next = new_node;
}

// Display the circular linked list
void display(struct Node *head)
{
    struct Node *temp = head;
    if (head != NULL)
    {
        do
        {
            printf("%d ", temp->data);
            temp = temp->next;
        } while (temp != head);
        printf("\n");
    }
    else
    {
        printf("List is empty.\n");
    }
}

// Delete a node with a specific value from the circular linked list
```

```c
void deleteNode(struct Node **head_ref, int key)
{
    if (*head_ref == NULL)
        return;

    struct Node *temp = *head_ref, *prev;

    // If the node to be deleted is the head
    if (temp->data == key && temp->next == *head_ref)
    {
        *head_ref = NULL;
        free(temp);
        return;
    }

    // If the node to be deleted is the head and the list has more than one node
    if (temp->data == key)
    {
        while (temp->next != *head_ref)
            temp = temp->next;
        temp->next = (*head_ref)->next;
        free(*head_ref);
        *head_ref = temp->next;
        return;
    }

    // If the node to be deleted is not the head
    prev = temp;
    while (temp->next != *head_ref && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }

    if (temp->data == key)
    {
        prev->next = temp->next;
        free(temp);
    }
}

void search(struct Node *head, int key)
{
    struct Node *temp = head;
    int pos = 0;

    if (head == NULL)
    {
        printf("List is empty.\n");
        return;
    }

    do
    {
        if (temp->data == key)
        {
            printf("Element %d found at position %d\n", key, pos);
            return;
        }
        temp = temp->next;
        pos++;
    } while (temp != head);

    printf("Element %d not found in the list\n", key);
}

int count(struct Node *head)
{
    int count = 0;
    struct Node *temp = head;

    if (head == NULL)
        return 0;

    do
    {
        count++;
        temp = temp->next;
    } while (temp != head);

    return count;
}

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

        printf("\nCircular 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;
    }

```