

SEM 3\Exp3\CSLL.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Node structure for the circular linked list
5  struct Node
6  {
7      int data;
8      struct Node *next;
9  };
10
11 // Insert a new node at the end of the circular linked list
12 void insert(struct Node **head_ref, int new_data)
13 {
14     struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
15     struct Node *temp = *head_ref;
16     new_node->data = new_data;
17     new_node->next = *head_ref;
18
19     if (*head_ref == NULL)
20     {
21         new_node->next = new_node;
22         *head_ref = new_node;
23         return;
24     }
25
26     while (temp->next != *head_ref)
27         temp = temp->next;
28
29     temp->next = new_node;
30 }
31
32 // Display the circular linked list
33 void display(struct Node *head)
34 {
35     struct Node *temp = head;
36     if (head != NULL)
37     {
38         do
39         {
40             printf("%d ", temp->data);
41             temp = temp->next;
42         } while (temp != head);
43         printf("\n");
44     }
45     else
46     {
47         printf("List is empty.\n");
48     }
49 }
50
51 // Delete a node with a specific value from the circular linked list
```

```
52 void deleteNode(struct Node **head_ref, int key)
53 {
54     if (*head_ref == NULL)
55         return;
56
57     struct Node *temp = *head_ref, *prev;
58
59     // If the node to be deleted is the head
60     if (temp->data == key && temp->next == *head_ref)
61     {
62         *head_ref = NULL;
63         free(temp);
64         return;
65     }
66
67     // If the node to be deleted is the head and the list has more than one node
68     if (temp->data == key)
69     {
70         while (temp->next != *head_ref)
71             temp = temp->next;
72         temp->next = (*head_ref)->next;
73         free(*head_ref);
74         *head_ref = temp->next;
75         return;
76     }
77
78     // If the node to be deleted is not the head
79     prev = temp;
80     while (temp->next != *head_ref && temp->data != key)
81     {
82         prev = temp;
83         temp = temp->next;
84     }
85
86     if (temp->data == key)
87     {
88         prev->next = temp->next;
89         free(temp);
90     }
91 }
92
93 void search(struct Node *head, int key)
94 {
95     struct Node *temp = head;
96     int pos = 0;
97
98     if (head == NULL)
99     {
100         printf("List is empty.\n");
101         return;
102     }
103
104     do
105     {
```

```
106         if (temp->data == key)
107         {
108             printf("Element %d found at position %d\n", key, pos);
109             return;
110         }
111         temp = temp->next;
112         pos++;
113     } while (temp != head);
114
115     printf("Element %d not found in the list\n", key);
116 }
117
118 int count(struct Node *head)
119 {
120     int count = 0;
121     struct Node *temp = head;
122
123     if (head == NULL)
124         return 0;
125
126     do
127     {
128         count++;
129         temp = temp->next;
130     } while (temp != head);
131
132     return count;
133 }
134
135 int main()
136 {
137     struct Node *head = NULL;
138     int choice, value, key;
139
140     printf("\nCircular Linked List Operations:\n");
141     printf("1. Insert\n");
142     printf("2. Display\n");
143     printf("3. Delete\n");
144     printf("4. Search\n");
145     printf("5. Count\n");
146     printf("6. Exit\n");
147
148     while (1)
149     {
150
151         printf("Enter your choice: ");
152         scanf("%d", &choice);
153
154         switch (choice)
155         {
156             case 1:
157                 printf("Enter the value to insert: ");
158                 scanf("%d", &value);
159                 insert(&head, value);
```

```
160         printf("\n");
161         break;
162     case 2:
163         display(head);
164         printf("\n");
165         break;
166     case 3:
167         printf("Enter the value to delete: ");
168         scanf("%d", &key);
169         deleteNode(&head, key);
170         printf("\n");
171         break;
172     case 4:
173         printf("Enter the value to search: ");
174         scanf("%d", &key);
175         search(head, key);
176         printf("\n");
177         break;
178     case 5:
179         printf("The number of nodes in the list: %d\n", count(head));
180         printf("\n");
181         break;
182     case 6:
183         exit(0);
184     default:
185         printf("Invalid choice!\n");
186     }
187 }
188
189 return 0;
190 }
191
```