

Folder SEM 3\Exp3

1 printable files

(file list disabled)

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 // Search for a specific value in the circular linked list
94 void search(struct Node *head, int key)
95 {
96     struct Node *temp = head;
97     int pos = 0;
98
99     if (head == NULL)
```

```
100     {
101         printf("List is empty.\n");
102         return;
103     }
104
105     do
106     {
107         if (temp->data == key)
108         {
109             printf("Element %d found at position %d\n", key, pos);
110             return;
111         }
112         temp = temp->next;
113         pos++;
114     } while (temp != head);
115
116     printf("Element %d not found in the list\n", key);
117 }
118
119 // Count the number of nodes in the circular linked list
120 int count(struct Node *head)
121 {
122     int count = 0;
123     struct Node *temp = head;
124
125     if (head == NULL)
126         return 0;
127
128     do
129     {
130         count++;
131         temp = temp->next;
132     } while (temp != head);
133
134     return count;
135 }
136
137 int main()
138 {
139     struct Node *head = NULL;
140     int choice, value, key;
141
142     while (1)
143     {
144         printf("\nCircular Linked List Operations:\n");
145         printf("1. Insert\n");
146         printf("2. Display\n");
147         printf("3. Delete\n");
148         printf("4. Search\n");
149         printf("5. Count\n");
150         printf("6. Exit\n");
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         break;
161     case 2:
162         display(head);
163         break;
164     case 3:
165         printf("Enter the value to delete: ");
166         scanf("%d", &key);
167         deleteNode(&head, key);
168         break;
169     case 4:
170         printf("Enter the value to search: ");
171         scanf("%d", &key);
172         search(head, key);
173         break;
174     case 5:
175         printf("The number of nodes in the list: %d\n", count(head));
176         break;
177     case 6:
178         exit(0);
179     default:
180         printf("Invalid choice!\n");
181     }
182 }
183
184 return 0;
185 }
186
```