

Folder SEM 3\Exp4

1 printable files

(file list disabled)

SEM 3\Exp4\CDLL.c

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

```
46     {
47         printf("%d ", temp→data);
48         temp = temp→next;
49     } while (temp ≠ head);
50     printf("\n");
51
52     printf("Traversal in reverse direction:\n");
53     temp = head→prev;
54     do
55     {
56         printf("%d ", temp→data);
57         temp = temp→prev;
58     } while (temp→next ≠ head);
59     printf("\n");
60 }
61
62 // Delete a node from the circular doubly linked list
63 void deleteNode(struct Node **head_ref, int key)
64 {
65     if (*head_ref == NULL)
66         return;
67
68     struct Node *current = *head_ref;
69
70     while (current→data ≠ key)
71     {
72         current = current→next;
73         if (current == *head_ref)
74         {
75             printf("Element %d not found in the list.\n", key);
76             return;
77         }
78     }
79
80     if (current→next == *head_ref && current→prev == *head_ref)
81     {
82         *head_ref = NULL;
83         free(current);
84         return;
85     }
86
87     if (current == *head_ref)
88     {
89         struct Node *last = (*head_ref)→prev;
90         *head_ref = current→next;
91         last→next = *head_ref;
92         (*head_ref)→prev = last;
93         free(current);
94         return;
95     }
96
97     current→prev→next = current→next;
98     current→next→prev = current→prev;
99
```

```
100     free(current);
101 }
102
103 // Search for a specific value in the circular doubly linked list
104 void search(struct Node *head, int key)
105 {
106     if (head == NULL)
107     {
108         printf("List is empty.\n");
109         return;
110     }
111
112     struct Node *temp = head;
113     int pos = 0;
114
115     do
116     {
117         if (temp->data == key)
118         {
119             printf("Element %d found at position %d\n", key, pos);
120             return;
121         }
122         temp = temp->next;
123         pos++;
124     } while (temp != head);
125
126     printf("Element %d not found in the list\n", key);
127 }
128
129 // Count the number of nodes in the circular doubly linked list
130 int count(struct Node *head)
131 {
132     if (head == NULL)
133         return 0;
134
135     struct Node *temp = head;
136     int count = 0;
137
138     do
139     {
140         count++;
141         temp = temp->next;
142     } while (temp != head);
143
144     return count;
145 }
146
147 int main()
148 {
149     struct Node *head = NULL;
150     int choice, value, key;
151
152     while (1)
153     {
```

```
154     printf("\nCircular Doubly Linked List Operations:\n");
155     printf("1. Insert\n");
156     printf("2. Display\n");
157     printf("3. Delete\n");
158     printf("4. Search\n");
159     printf("5. Count\n");
160     printf("6. Exit\n");
161     printf("Enter your choice: ");
162     scanf("%d", &choice);
163
164     switch (choice)
165     {
166     case 1:
167         printf("Enter the value to insert: ");
168         scanf("%d", &value);
169         insert(&head, value);
170         break;
171     case 2:
172         display(head);
173         break;
174     case 3:
175         printf("Enter the value to delete: ");
176         scanf("%d", &key);
177         deleteNode(&head, key);
178         break;
179     case 4:
180         printf("Enter the value to search: ");
181         scanf("%d", &key);
182         search(head, key);
183         break;
184     case 5:
185         printf("The number of nodes in the list: %d\n", count(head));
186         break;
187     case 6:
188         exit(0);
189     default:
190         printf("Invalid choice!\n");
191     }
192 }
193
194 return 0;
195 }
196
```