

SEM 3\Exp2\DLL_implementation.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Node structure for the doubly linked list
5  struct Node
6  {
7      int data;
8      struct Node *prev;
9      struct Node *next;
10 };
11
12 // Insert at the end of the 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     struct Node *last = *head_ref;
17     new_node->data = new_data;
18     new_node->next = NULL;
19
20     if (*head_ref == NULL)
21     {
22         new_node->prev = NULL;
23         *head_ref = new_node;
24         return;
25     }
26
27     while (last->next != NULL)
28         last = last->next;
29
30     last->next = new_node;
31     new_node->prev = last;
32 }
33
34 // Display the doubly linked list
35 void display(struct Node *node)
36 {
37     struct Node *last;
38     printf("Traversal in forward direction:\n");
39     while (node != NULL)
40     {
41         printf("%d ", node->data);
42         last = node;
43         node = node->next;
44     }
45
46     printf("\nTraversal in reverse direction:\n");
47     while (last != NULL)
48     {
49         printf("%d ", last->data);
50         last = last->prev;
51     }
```

```
52     printf("\n");
53 }
54
55 // Delete a node from the doubly linked list
56 void deleteNode(struct Node **head_ref, int key)
57 {
58     struct Node *temp = *head_ref;
59
60     if (*head_ref == NULL)
61         return;
62
63     while (temp != NULL && temp->data != key)
64         temp = temp->next;
65
66     if (temp == NULL)
67         return;
68
69     if (*head_ref == temp)
70         *head_ref = temp->next;
71
72     if (temp->next != NULL)
73         temp->next->prev = temp->prev;
74
75     if (temp->prev != NULL)
76         temp->prev->next = temp->next;
77
78     free(temp);
79 }
80
81 // Search for a key in the doubly linked list
82 void search(struct Node *head, int key)
83 {
84     struct Node *temp = head;
85     int pos = 0;
86     while (temp != NULL)
87     {
88         if (temp->data == key)
89         {
90             printf("Element %d found at position %d\n", key, pos);
91             return;
92         }
93         temp = temp->next;
94         pos++;
95     }
96     printf("Element %d not found in the list\n", key);
97 }
98
99 // Count the number of nodes in the doubly linked list
100 int count(struct Node *head)
101 {
102     int count = 0;
103     struct Node *temp = head;
104     while (temp != NULL)
105     {
```

```
106         count++;
107         temp = temp->next;
108     }
109     return count;
110 }
111
112 int main()
113 {
114     struct Node *head = NULL;
115     int choice, value, key;
116
117     printf("\nDoubly Linked List Operations:\n");
118     printf("1. Insert\n");
119     printf("2. Display\n");
120     printf("3. Delete\n");
121     printf("4. Search\n");
122     printf("5. Count\n");
123     printf("6. Exit\n");
124
125     while (1)
126     {
127         printf("Enter your choice: ");
128         scanf("%d", &choice);
129
130         switch (choice)
131         {
132             case 1:
133                 printf("Enter the value to insert: ");
134                 scanf("%d", &value);
135                 insert(&head, value);
136                 printf("\n");
137                 break;
138             case 2:
139                 display(head);
140                 printf("\n");
141                 break;
142             case 3:
143                 printf("Enter the value to delete: ");
144                 scanf("%d", &key);
145                 deleteNode(&head, key);
146                 printf("\n");
147                 break;
148             case 4:
149                 printf("Enter the value to search: ");
150                 scanf("%d", &key);
151                 search(head, key);
152                 printf("\n");
153                 break;
154             case 5:
155                 printf("The number of nodes in the list: %d\n", count(head));
156                 printf("\n");
157                 break;
158             case 6:
159                 exit(0);
```

```
160         default:
161             printf("Invalid choice!\n");
162         }
163     }
164
165     return 0;
166 }
167
```