

## SEM 3\Exp6\Queue\_LL.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Node structure for the linked list
5  struct Node
6  {
7      int data;
8      struct Node *next;
9  };
10
11 // Queue structure using linked lists
12 struct QueueLinkedList
13 {
14     struct Node *front;
15     struct Node *rear;
16 };
17
18 // Function to create a queue
19 struct QueueLinkedList *createQueue()
20 {
21     struct QueueLinkedList *queue = (struct QueueLinkedList *)malloc(sizeof(struct
QueueLinkedList));
22     queue->front = queue->rear = NULL; // Initialize front and rear
23     return queue;
24 }
25
26 // Check if the queue is empty
27 int isEmpty(struct QueueLinkedList *queue)
28 {
29     return queue->front == NULL;
30 }
31
32 // Enqueue an element into the queue
33 void enqueue(struct QueueLinkedList *queue, int value)
34 {
35     struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
36     new_node->data = value;
37     new_node->next = NULL;
38
39     if (isEmpty(queue))
40     {
41         queue->front = queue->rear = new_node; // First node
42         printf("%d enqueued to queue\n", value);
43         return;
44     }
45
46     queue->rear->next = new_node; // Add new node at the end
47     queue->rear = new_node;      // Update the rear pointer
48     printf("%d enqueued to queue\n", value);
49 }
50
51 // Dequeue an element from the queue
```

```
52 int dequeue(struct QueueLinkedList *queue)
53 {
54     if (isEmpty(queue))
55     {
56         printf("Queue underflow!\n");
57         return -1;
58     }
59     struct Node *temp = queue->front;
60     int dequeued_value = temp->data;
61     queue->front = queue->front->next;
62
63     // If the front becomes NULL, set rear to NULL as well
64     if (queue->front == NULL)
65     {
66         queue->rear = NULL;
67     }
68
69     free(temp);
70     return dequeued_value;
71 }
72
73 // Peek at the front element of the queue
74 int peek(struct QueueLinkedList *queue)
75 {
76     if (isEmpty(queue))
77     {
78         printf("Queue is empty!\n");
79         return -1;
80     }
81     return queue->front->data;
82 }
83
84 // Display the queue
85 void display(struct QueueLinkedList *queue)
86 {
87     if (isEmpty(queue))
88     {
89         printf("Queue is empty!\n");
90         return;
91     }
92     struct Node *temp = queue->front;
93     printf("Queue elements: ");
94     while (temp != NULL)
95     {
96         printf("%d ", temp->data);
97         temp = temp->next;
98     }
99     printf("\n");
100 }
101
102 int main()
103 {
104     struct QueueLinkedList *queue = createQueue();
105     int choice, value;
```

```
106
107     printf("\nQueue Operations (Linked List Implementation):\n");
108     printf("1. Enqueue\n");
109     printf("2. Dequeue\n");
110     printf("3. Peek\n");
111     printf("4. Display\n");
112     printf("5. Exit\n");
113
114     while (1)
115     {
116
117         printf("Enter your choice: ");
118         scanf("%d", &choice);
119
120         switch (choice)
121         {
122             case 1:
123                 printf("Enter the value to enqueue: ");
124                 scanf("%d", &value);
125                 enqueue(queue, value);
126                 printf("\n");
127                 break;
128             case 2:
129                 value = dequeue(queue);
130                 if (value != -1)
131                     printf("Dequeued value: %d\n", value);
132
133                 printf("\n");
134                 break;
135             case 3:
136                 value = peek(queue);
137                 if (value != -1)
138                     printf("Front value: %d\n", value);
139
140                 printf("\n");
141                 break;
142             case 4:
143                 display(queue);
144                 printf("\n");
145                 break;
146             case 5:
147                 // Free linked list nodes (cleanup)
148                 while (!isEmpty(queue))
149                 {
150                     dequeue(queue);
151                 }
152                 free(queue);
153                 exit(0);
154             default:
155                 printf("Invalid choice!\n");
156         }
157     }
158
159     return 0;
```

```
160 | }  
161 |
```