Experiments\stack_linkedlist.c

```c
1   // C program to implement the queue data structure using
2   // linked list
3   #include <limits.h>
4   #include <stdio.h>
5   #include <stdlib.h>
6
7   // Node structure representing a single node in the linked
8   // list
9   typedef struct Node {
10      int data;
11      struct Node* next;
12  } Node;
13
14  // Function to create a new node
15  Node* createNode(int new_data)
16  {
17      Node* new_node = (Node*)malloc(sizeof(Node));
18      new_node->data = new_data;
19      new_node->next = NULL;
20      return new_node;
21  }
22
23  // Structure to implement queue operations using a linked
24  // list
25  typedef struct Queue {
26
27      // Pointer to the front and the rear of the linked list
28      Node *front, *rear;
29  } Queue;
30
31  // Function to create a queue
32  Queue* createQueue()
33  {
34      Queue* q = (Queue*)malloc(sizeof(Queue));
35      q->front = q->rear = NULL;
36      return q;
37  }
38
39  // Function to check if the queue is empty
40  int isEmpty(Queue* q)
41  {
42
43      // If the front and rear are null, then the queue is
44      // empty, otherwise it's not
45      if (q->front == NULL && q->rear == NULL) {
46          return 1;
47      }
48      return 0;
```

```c
49  }
50
51  // Function to add an element to the queue
52  void enqueue(Queue* q, int new_data)
53  {
54
55      // Create a new linked list node
56      Node* new_node = createNode(new_data);
57
58      // If queue is empty, the new node is both the front
59      // and rear
60      if (q->rear == NULL) {
61          q->front = q->rear = new_node;
62          return;
63      }
64
65      // Add the new node at the end of the queue and
66      // change rear
67      q->rear->next = new_node;
68      q->rear = new_node;
69  }
70
71  // Function to remove an element from the queue
72  void dequeue(Queue* q)
73  {
74
75      // If queue is empty, return
76      if (isEmpty(q)) {
77          printf("Queue Underflow\n");
78          return;
79      }
80
81      // Store previous front and move front one node
82      // ahead
83      Node* temp = q->front;
84      q->front = q->front->next;
85
86      // If front becomes null, then change rear also
87      // to null
88      if (q->front == NULL)
89          q->rear = NULL;
90
91      // Deallocate memory of the old front node
92      free(temp);
93  }
94
95  // Function to get the front element of the queue
96  int getFront(Queue* q)
97  {
98
```

```c
 99        // Checking if the queue is empty
100        if (isEmpty(q)) {
101            printf("Queue is empty\n");
102            return INT_MIN;
103        }
104        return q->front->data;
105    }
106
107    // Function to get the rear element of the queue
108    int getRear(Queue* q)
109    {
110
111        // Checking if the queue is empty
112        if (isEmpty(q)) {
113            printf("Queue is empty\n");
114            return INT_MIN;
115        }
116        return q->rear->data;
117    }
118
119    // Driver code
120    int main()
121    {
122        Queue* q = createQueue();
123
124        // Enqueue elements into the queue
125        enqueue(q, 10);
126        enqueue(q, 20);
127
128          printf("Queue Front: %d\n", getFront(q));
129        printf("Queue Rear: %d\n", getRear(q));
130
131        // Dequeue elements from the queue
132        dequeue(q);
133        dequeue(q);
134
135
136        // Enqueue more elements into the queue
137        enqueue(q, 30);
138        enqueue(q, 40);
139        enqueue(q, 50);
140
141        // Dequeue an element from the queue
142        dequeue(q);
143
144        printf("Queue Front: %d\n", getFront(q));
145        printf("Queue Rear: %d\n", getRear(q));
146
147        return 0;
148    }
```

```
149   /*
150   Output:PS C:\Users\gagan\Documents\Data_Structure_And_Algorithm> cd
      "c:\Users\gagan\Documents\Data_Structure_And_Algorithm\Experiments\" ; if ($?) { gcc
      stack_linkedlist.c -o stack_linkedlist } ; if ($?) { .\stack_linkedlist }
151   Queue Front: 10
152   Queue Rear: 20
153   Queue Front: 40
154   Queue Rear: 50
155   */
```