

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct Node{
5      int data;
6      struct Node *next;
7  }node;
8
9  typedef struct DeQueue{
10     node *front;
11     node *rear;
12 }dequeue;
13
14 dequeue create(){
15     dequeue dq = {NULL, NULL};
16     return dq;
17 }
18
19 int rear(dequeue dq){
20     return dq.rear->data;
21 }
22
23 int front(dequeue dq){
24     return dq.front->data;
25 }
26 int len(dequeue dq){
27     if(dq.front == NULL){
28         return 0;
29     }
30     int count = 1;
31     while(dq.rear != dq.front){
32         count++;
33         dq.rear = dq.rear->next;
34     }
35     return count;
36 }
37
38 int isEmpty(dequeue dq){
39     if(dq.front){
40         return 0;
41     }
42     return 1;
43 }
44
45 void enq_rear(dequeue *dq, int new_data){
46     node *new_node = (node*)malloc(sizeof(node));
47     new_node->data = new_data;
48     new_node->next = dq->rear;
49     if(dq->rear == NULL){
50         dq->front = new_node;
51     }
52     dq->rear = new_node;
53 }
54
55 void enq_front(dequeue *dq, int new_data){
56     node *new_node = (node*)malloc(sizeof(node));
57     new_node->data = new_data;
58     new_node->next = NULL;
59     dq->front->next = new_node;
60     if(dq->front == NULL){
61         dq->rear = new_node;
62     }
63     dq->front = new_node;
64 }
```

```

65
66 int deq_rear(dequeue *dq){
67     if(!dq->rear){
68         printf("UnderFlow.\n");
69         return -1;
70     }
71     node *temp = dq->rear;
72     dq->rear = dq->rear->next;
73     int data = temp->data;
74     free(temp);
75     return data;
76 }
77
78 int deq_front(dequeue *dq){
79     if(dq->front == NULL){
80         printf("Underflow.\n");
81         return -1;
82     }
83     int data;
84     node *temp = dq->rear;
85     if(dq->rear == dq->front){
86         data = temp->data;
87         *dq = create();
88         free(temp);
89         return data;
90     }
91     while(temp->next != dq->front){
92         temp = temp->next;
93     }
94     temp->next = NULL;
95     node *next = dq->front;
96     dq->front = temp;
97     data = next->data;
98     free(next);
99     return data;
100 }
101
102 void display(dequeue dq){
103     if(dq.rear == NULL){
104         printf("The Queue is Empty.");
105     }
106     node *temp = dq.rear;
107     while(temp != NULL){
108         printf("%d",temp->data);
109         temp = temp->next;
110         if(temp != NULL){
111             printf(" -> ");
112         }
113     }
114     printf("\n");
115 }
116
117 void main(){
118     int data;
119     int op;
120     dequeue dq = create();
121     while(1){
122         printf("1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8.
Front, 9. Exit.\nEnter Your Choice: ");
123         scanf("%d",&op);
124         switch(op){
125             case 1:
126                 printf("Enter the element: ");
127                 scanf("%d",&data);
128                 enq_rear(&dq, data);
129                 break;
130             case 2:

```

```

131         printf("Enter the element: ");
132         scanf("%d",&data);
133         enq_front(&dq, data);
134         break;
135     case 3:
136         printf("Removed Rear %d\n",deq_rear(&dq));
137         break;
138     case 4:
139         printf("Removed Front %d\n",deq_front(&dq));
140         break;
141     case 5:
142         display(dq);
143         break;
144     case 6:
145         printf("The len of this Queue is %d\n",len(dq));
146         break;
147     case 7:
148         printf("The Rear of this Queue is %d\n",rear(dq));
149         break;
150     case 8:
151         printf("The Front of this Queue is %d\n",front(dq));
152         break;
153     case 9:
154         printf("Oops..");
155         return;
156     default:
157         printf("Wrong Input.\n");
158 }
159 }
160 }
161
162 /*OUTPUT
163 PS S:\Workspace\CollegeWork\DataStructure> gcc .\dequeue-using-linkd-list.c
164 PS S:\Workspace\CollegeWork\DataStructure> ./a
165 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
166 Enter Your Choice: 1
167 Enter the element: 12
168 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
169 Enter Your Choice: 1
170 Enter the element: 13
171 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
172 Enter Your Choice: 2
173 Enter the element: 23
174 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
175 Enter Your Choice: 5
176 13 -> 12 -> 23
177 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
178 Enter Your Choice: 2
179 Enter the element: 34
180 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
181 Enter Your Choice: 5
182 13 -> 12 -> 23 -> 34
183 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
184 Enter Your Choice: 3
185 Removed Rear 13
186 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
187 Enter Your Choice: 4
188 Removed Front 34
189 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
190 Enter Your Choice: 5
191 12 -> 23
192 1. enqueueR, 2. enqueueF, 3. dequeueR, 4. dequeueF, 5. Display, 6. Size, 7. Rear, 8. Front, 9. Exit.
193 Enter Your Choice: 9
194 Oops..
195 PS S:\Workspace\CollegeWork\DataStructure>
196 */

```