

Experiment 4

CODE:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define SIZE 5
5
6  typedef struct {
7      int items[SIZE];
8      int front;
9      int rear;
10 } CircularQueue;
11
12 void initQueue(CircularQueue *q) {
13     q->front = -1;
14     q->rear = -1;
15 }
16
17 int isFull(CircularQueue *q) {
18     return (q->front == (q->rear + 1) % SIZE);
19 }
20
21 int isEmpty(CircularQueue *q) {
22     return (q->front == -1);
23 }
24
25 void enqueue(CircularQueue *q, int value) {
26     if (isFull(q)) {
27         printf("Queue is full!\n");
28     } else {
29         if (q->front == -1) q->front = 0;
30         q->rear = (q->rear + 1) % SIZE;
31         q->items[q->rear] = value;
32         printf("Inserted %d\n", value);
33     }
34 }
35
36 int dequeue(CircularQueue *q) {
37     int element;
38     if (isEmpty(q)) {
39         printf("Queue is empty!\n");
40         return -1;
41     } else {
42         element = q->items[q->front];
43         if (q->front == q->rear) {
44             q->front = -1;
45             q->rear = -1;
46         } else {
47             q->front = (q->front + 1) % SIZE;
48         }
49         return element;
50     }
51 }
52
53 void display(CircularQueue *q) {
54     int i;
55     if (isEmpty(q)) {
56         printf("Queue is empty!\n");
57     } else {
58         printf("Queue elements are:\n");
59         for (i = q->front; i != q->rear; i = (i + 1) % SIZE) {
60             printf("%d ", q->items[i]);
61         }
62         printf("%d\n", q->items[i]);
63     }
64 }
65
66 int getFront(CircularQueue *q) {
67     if (isEmpty(q)) {
68         printf("Queue is empty!\n");
69         return -1;
70     }
}
```

```

71     return q->items[q->front];
72 }
73
74 int getRear(CircularQueue *q) {
75     if (isEmpty(q)) {
76         printf("Queue is empty!\n");
77         return -1;
78     }
79     return q->items[q->rear];
80 }
81
82 int main() {
83     CircularQueue q;
84     initQueue(&q);
85     int choice, value;
86
87     do {
88         printf("\nMenu:\n");
89         printf("1. Enqueue\n");
90         printf("2. Dequeue\n");
91         printf("3. Display\n");
92         printf("4. Check if Queue is Full\n");
93         printf("5. Check if Queue is Empty\n");
94         printf("6. Get Front\n");
95         printf("7. Get Rear\n");
96         printf("8. Exit\n");
97         printf("Enter your choice: ");
98         scanf("%d", &choice);
99
100        switch (choice) {
101            case 1:
102                printf("Enter value to enqueue: ");
103                scanf("%d", &value);
104                enqueue(&q, value);
105                display(&q);
106                break;
107            case 2:
108                value = dequeue(&q);


---


109                if (value != -1) {
110                    printf("Dequeued value: %d\n", value);
111                }
112                display(&q);
113                break;
114            case 3:
115                display(&q);
116                break;
117            case 4:
118                if (isFull(&q)) {
119                    printf("Queue is full!\n");
120                } else {
121                    printf("Queue is not full.\n");
122                }
123                break;
124            case 5:
125                if (isEmpty(&q)) {
126                    printf("Queue is empty!\n");
127                } else {
128                    printf("Queue is not empty.\n");
129                }
130                break;
131            case 6:
132                value = getFront(&q);
133                if (value != -1) {
134                    printf("Front value: %d\n", value);
135                }
136                display(&q);
137                break;
138            case 7:
139                value = getRear(&q);
140                if (value != -1) {
141                    printf("Rear value: %d\n", value);
142                }
143                display(&q);
144                break;
145            case 8:
146

```

```

146     :   :   :   printf("Exiting...\n");
147     :   :   :   break;
148     :   :   default:
149     :   :   :   printf("Invalid choice! Please try again.\n");
150     :   }
151     } while (choice != 8);
152
153     return 0;
154 }

```

OUTPUT:

Menu:

1. Enqueue
 2. Dequeue
 3. Display
 4. Check if Queue is Full
 5. Check if Queue is Empty
 6. Get Front
 7. Get Rear
 8. Exit
 Enter your choice: 1
 Enter value to enqueue: 34
 Inserted 34
 Queue elements are:
 34

Menu:

1. Enqueue
 2. Dequeue
 3. Display
 4. Check if Queue is Full
 5. Check if Queue is Empty
 6. Get Front
 7. Get Rear
 8. Exit
 Enter your choice: 1
 Enter value to enqueue: 45
 Inserted 45
 Queue elements are:
 34 45

Menu:

1. Enqueue
 2. Dequeue
 3. Display
 4. Check if Queue is Full
 5. Check if Queue is Empty
 6. Get Front
 7. Get Rear
 8. Exit
 Enter your choice: 1
 Enter value to enqueue: 43
 Inserted 43
 Queue elements are:
 34 45 43

Menu:

1. Enqueue
 2. Dequeue
 3. Display
 4. Check if Queue is Full
 5. Check if Queue is Empty
 6. Get Front
 7. Get Rear
 8. Exit
 Enter your choice: 2
 Dequeued value: 34
 Queue elements are:
 45 43

Menu:

1. Enqueue
2. Dequeue
3. Display
4. Check if Queue is Full
5. Check if Queue is Empty
6. Get Front
7. Get Rear
8. Exit

Enter your choice: 6

Front value: 45

Queue elements are:

45 43

Menu:

1. Enqueue
2. Dequeue
3. Display
4. Check if Queue is Full
5. Check if Queue is Empty
6. Get Front
7. Get Rear
8. Exit

Enter your choice: 7

Rear value: 43

Queue elements are:

45 43

Menu:

1. Enqueue
2. Dequeue
3. Display
4. Check if Queue is Full
5. Check if Queue is Empty
6. Get Front

7. Get Rear

8. Exit

Enter your choice: 5

Queue is not empty.

Menu:

1. Enqueue
2. Dequeue
3. Display
4. Check if Queue is Full
5. Check if Queue is Empty
6. Get Front
7. Get Rear
8. Exit

Enter your choice: 4

Queue is not full.

Menu:

1. Enqueue
2. Dequeue
3. Display
4. Check if Queue is Full
5. Check if Queue is Empty
6. Get Front
7. Get Rear
8. Exit

Enter your choice: 8

Exiting...

=== Code Execution Successful ===