

## Experiment 3

### CODE:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MAX 5
5
6  typedef struct {
7      int items[MAX];
8      int front, rear;
9  } Queue;
10
11 void initQueue(Queue *q) {
12     q->front = -1;
13     q->rear = -1;
14 }
15
16 int isQueueEmpty(Queue *q) {
17     return q->front == -1;
18 }
19
20 int isQueueFull(Queue *q) {
21     return (q->rear + 1) % MAX == q->front;
22 }
23
24 void enqueue(Queue *q, int item) {
25     if (isQueueFull(q)) {
26         printf("Queue is full. Cannot enqueue %d.\n", item);
27         return;
28     }
29     if (isQueueEmpty(q)) {
30         q->front = 0;
31     }
32     q->rear = (q->rear + 1) % MAX;
33     q->items[q->rear] = item;
34     printf("Enqueued %d\n", item);
35 }
36
37 int dequeue(Queue *q) {
38     if (isQueueEmpty(q)) {
39         printf("Queue is empty. Cannot dequeue.\n");
40         return -1;
41     }
42     int item = q->items[q->front];
43     if (q->front == q->rear) {
44         initQueue(q);
45     } else {
46         q->front = (q->front + 1) % MAX;
47     }
48     return item;
49 }
50
51 int getFront(Queue *q) {
52     if (isQueueEmpty(q)) {
53         printf("Queue is empty.\n");
54         return -1;
55     }
56     return q->items[q->front];
57 }
58
59 int getRear(Queue *q) {
60     if (isQueueEmpty(q)) {
61         printf("Queue is empty.\n");
62         return -1;
63     }
64     return q->items[q->rear];
65 }
66
```

```

67- void displayQueue(Queue *q) {
68-     if (isEmpty(q)) {
69         printf("Queue is empty.\n");
70         return;
71     }
72     printf("Queue contents: ");
73     int i = q->front;
74-     while (1) {
75         printf("%d ", q->items[i]);
76         if (i == q->rear) break;
77         i = (i + 1) % MAX;
78     }
79     printf("\n");
80 }
81
82- int main() {
83     Queue q;
84     initQueue(&q);
85     int choice, item;
86
87-     do {
88         printf("\nQueue Menu:\n");
89         printf("1. Enqueue\n");
90         printf("2. Dequeue\n");
91         printf("3. Get Front\n");
92         printf("4. Get Rear\n");
93         printf("5. Check if Queue is Empty\n");
94         printf("6. Check if Queue is Full\n");
95         printf("7. Display Queue\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 the item to enqueue: ");
103                scanf("%d", &item);
104                enqueue(&q, item);
105                displayQueue(&q);
106                break;
107            case 2:
108                item = dequeue(&q);
109-                if (item != -1) {
110                    printf("Dequeued %d\n", item);
111                }
112                displayQueue(&q);
113                break;
114            case 3:
115                item = getFront(&q);
116-                if (item != -1) {
117                    printf("Front item is %d\n", item);
118                }
119                break;
120            case 4:
121                item = getRear(&q);
122-                if (item != -1) {
123                    printf("Rear item is %d\n", item);
124                }
125                break;
126            case 5:
127-                if (isEmpty(&q)) {
128                    printf("Queue is empty.\n");
129-                } else {
130                    printf("Queue is not empty.\n");
131                }
132                break;
133            case 6:
134-                if (isQueueFull(&q)) {
135                    printf("Queue is full.\n");
136-                } else {
137                    printf("Queue is not full.\n");
138                }
139                break;

```

```

140      :      :      case 7:
141      :      :      :      displayQueue(&q);
142      :      :      :      break;
143      :      :      case 8:
144      :      :      :      printf("Exiting...\n");
145      :      :      :      break;
146      :      :      default:
147      :      :      :      printf("Invalid choice! Please try again.\n");
148      :      :      :      break;
149      :      :      }
150      :      :      } while (choice != 8);
151
152      return 0;
153  }
```

## **OUTPUT:**

Queue Menu:

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

Enter your choice: 1

Enter the item to enqueue: 85

Enqueued 85

Queue contents: 85

Queue Menu:

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

Enter your choice: 1

Enter the item to enqueue: 58

Enqueued 58

Queue contents: 85 58

Queue Menu:

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

Enter your choice: 1

Enter the item to enqueue: 98

Enqueued 98

Queue contents: 85 58 98

Queue Menu:

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

8. Exit

Enter your choice: 2

Dequeued 85

Queue contents: 58 98

Queue Menu:

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

Enter your choice: 3

Front item is 58

Queue Menu:

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

Enter your choice: 4

Rear item is 98

Queue Menu:

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

7. Display Queue

8. Exit

Enter your choice: 6

Queue is not full.

Queue Menu:

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

Enter your choice: 2

Dequeued 58

Queue contents: 98

Queue Menu:

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

Enter your choice: 2

Dequeued 98

Queue is empty.

Queue Menu:

1. Enqueue
2. Dequeue
3. Get Front
4. Get Rear

5. Check if Queue is Empty
6. Check if Queue is Full
7. Display Queue
8. Exit

Enter your choice: 5

Queue is empty.

Queue Menu:

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

Enter your choice: 1

Enter the item to enqueue: 85

Enqueued 85

Queue contents: 85

Queue Menu:

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

Enter your choice: 1

Enter the item to enqueue: 89

Enqueued 89

Queue contents: 85 89

Queue Menu:

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

Enter your choice: 7

Queue contents: 85 89 75

Queue Menu:

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

Enter your choice: 8

Exiting...

=== Code Execution Successful ===