Experiment 4

CODE:

```c
#include <stdio.h>
#include <stdlib.h>

#define SIZE 5

typedef struct {
    int items[SIZE];
    int front;
    int rear;
} CircularQueue;

void initQueue(CircularQueue *q) {
    q->front = -1;
    q->rear = -1;
}

int isFull(CircularQueue *q) {
    return (q->front == (q->rear + 1) % SIZE);
}

int isEmpty(CircularQueue *q) {
    return (q->front == -1);
}

void enqueue(CircularQueue *q, int value) {
    if (isFull(q)) {
        printf("Queue is full!\n");
    } else {
        if (q->front == -1) q->front = 0;
        q->rear = (q->rear + 1) % SIZE;
        q->items[q->rear] = value;
        printf("Inserted %d\n", value);
    }
}

int dequeue(CircularQueue *q) {
    int element;
    if (isEmpty(q)) {
        printf("Queue is empty!\n");
        return -1;
    } else {
        element = q->items[q->front];
        if (q->front == q->rear) {
            q->front = -1;
            q->rear = -1;
        } else {
            q->front = (q->front + 1) % SIZE;
        }
        return element;
    }
}

void display(CircularQueue *q) {
    int i;
    if (isEmpty(q)) {
        printf("Queue is empty!\n");
    } else {
        printf("Queue elements are:\n");
        for (i = q->front; i != q->rear; i = (i + 1) % SIZE) {
            printf("%d ", q->items[i]);
        }
        printf("%d\n", q->items[i]);
    }
}

int getFront(CircularQueue *q) {
    if (isEmpty(q)) {
        printf("Queue is empty!\n");
        return -1;
    }
    return q->items[q->front];
}

int getRear(CircularQueue *q) {
    if (isEmpty(q)) {
        printf("Queue is empty!\n");
        return -1;
    }
    return q->items[q->rear];
}

int main() {
    CircularQueue q;
    initQueue(&q);
    int choice, value;

    do {
        printf("\nMenu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Check if Queue is Full\n");
        printf("5. Check if Queue is Empty\n");
        printf("6. Get Front\n");
        printf("7. Get Rear\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(&q, value);
                display(&q);
                break;
            case 2:
                value = dequeue(&q);
                if (value != -1) {
                    printf("Dequeued value: %d\n", value);
                }
                display(&q);
                break;
            case 3:
                display(&q);
                break;
            case 4:
                if (isFull(&q)) {
                    printf("Queue is full!\n");
                } else {
                    printf("Queue is not full.\n");
                }
                break;
            case 5:
                if (isEmpty(&q)) {
                    printf("Queue is empty!\n");
                } else {
                    printf("Queue is not empty.\n");
                }
                break;
            case 6:
                value = getFront(&q);
                if (value != -1) {
                    printf("Front value: %d\n", value);
                }
                display(&q);
                break;
            case 7:
                value = getRear(&q);
                if (value != -1) {
                    printf("Rear value: %d\n", value);
                }
                display(&q);
                break;
            case 8:

                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 8);

    return 0;
}
```

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 ===
```