**Code: -**

class CircularQueue:

    def \_\_init\_\_(self, capacity):

        self.capacity = capacity  # Size of the queue

        self.queue = [None] \* capacity  # Array to hold the queue elements

        self.front = -1  # Front pointer, initialized to -1 indicating the queue is empty

        self.rear = -1  # Rear pointer, initialized to -1 indicating the queue is empty

    def is\_empty(self):

        return self.front == -1  # Queue is empty if front is -1

    def is\_full(self):

        return (self.rear + 1) % self.capacity == self.front  # Queue is full if (rear + 1) % capacity == front

    def enqueue(self, value):

        if self.is\_full():

            print("Queue is full!")

            return

        if self.is\_empty():

            self.front = self.rear = 0  # When the first element is inserted

        else:

            self.rear = (self.rear + 1) % self.capacity  # Move rear to the next position circularly

        self.queue[self.rear] = value  # Insert the value at the rear

    def dequeue(self):

        if self.is\_empty():

            print("Queue is empty!")

            return None

        dequeued\_value = self.queue[self.front]

        if self.front == self.rear:  # If the front equals rear, reset the queue (only one element was in the queue)

            self.front = self.rear = -1

        else:

            self.front = (self.front + 1) % self.capacity  # Move front to the next position circularly

        return dequeued\_value

    def peek(self):

        if self.is\_empty():

            print("Queue is empty!")

            return None

        return self.queue[self.front]

    def display(self):

        if self.is\_empty():

            print("Queue is empty!")

            return

        print("Queue elements:")

        i = self.front

        while True:

            print(self.queue[i], end=" ")

            if i == self.rear:

                break

            i = (i + 1) % self.capacity

        print()  # For a clean line break

# Example usage

queue = CircularQueue(5)

queue.enqueue(10)

queue.enqueue(20)

queue.enqueue(30)

queue.enqueue(40)

queue.enqueue(50)

print("Initial Queue:")

queue.display()

print("\nDequeue operation:", queue.dequeue())

print("\nQueue after Dequeue:")

queue.display()

queue.enqueue(60)

print("\nQueue after Enqueue 60:")

queue.display()

queue.enqueue(70)

print("\nQueue after Enqueue 70:")

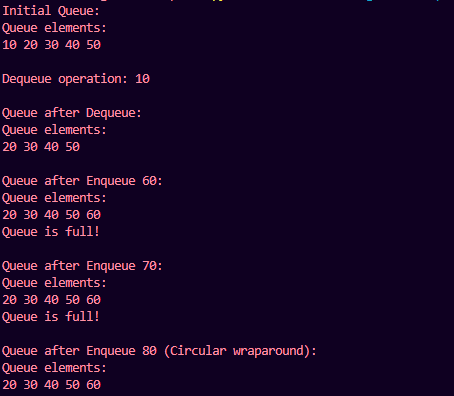
queue.display()

queue.enqueue(80)

print("\nQueue after Enqueue 80 (Circular wraparound):")

queue.display()

**OUTPUT:**

****