```
#Problem 1, 2, 3
print("\nProblem 1, 2, 3")

class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self,item):
        self.queue.append(item)
        print(f"Enqueued: {item}")

    def dequeue(self):
        if not self.is_empty():
            item = self.queue.pop(0)
            print(f"Dequeued:{item}")
            return item
        else:
            print("Queue is empty, cannot dequeue.")

    def display(self):
        print(self.queue)

    def is_empty(self):
        return len(self.queue) == 0

    def size(self):
        return len(self.queue)

queue1 = Queue()
```

```
print("\nEqueuing items:")
queue1.enqueue(1)
queue1.enqueue(2)
queue1.enqueue(3)

print("\nDisplaying queue:")
queue1.display()

print("\nDequeuing items:")
queue1.dequeue()
queue1.dequeue()

print("\nDisplaying queue after dequeuing:")
queue1.display()

print("\nTesting dequeue on empty queue:")
print(queue1.is_empty())

print("\nQueue size:")
print(queue1.size())

#Problem 4, 5, 6
print("\nProblem 4, 5, 6")

class CircularQueue:
  def __init__(self, size):
    self.size = size
    self.queue = [None] * size
```

```
    self.front = -1
    self.rear = -1

def  is_empty(self):
    return self.front == -1

def is_full(self):
    return (self.rear + 1) % self.size == self.front

def enqueue(self, element):
    if self.is_full():
        print("Queue is full! Cannot enqueue.")
        return
    if self.is_empty():
        self.front = 0
    self.rear = (self.rear + 1) % self.size
    self.queue[self.rear] = element

def dequeue(self):
    if self.is_empty():
        print("Queue is empty! Cannot dequeue.")
        return None
    element = self.queue[self.front]
    if self.front == self.rear:
        self.front = -1
        self.rear = -1
    else:
        self.front = (self.front + 1) % self.size
    return element
```

```
    def peek(self):
        if self.is_empty():
            print("Queue is empty! No element to peek.")
            return None
        return self.queue[self.front]

    def display(self):
        if self.is_empty():
            print("Queue is empty!")
            return
        print("Queue elements:")
        if self.rear >= self.front:
            print(self.queue[self.front: self.rear + 1])
        else:
            print(self.queue[self.front: self.size] + self.queue[0: self.rear + 1])

queue2 = CircularQueue(5)


print("\nEnqueuing elements 10, 20, 30:")
queue2.enqueue(10)
queue2.enqueue(20)
queue2.enqueue(30)
queue2.display()

print(f"Front element: {queue2.peek()}")
dequeued_element = queue2.dequeue()
print(f"Dequeued element: {dequeued_element}")
```

```
queue2.display()



queue2.enqueue(40)
queue2.enqueue(50)
queue2.enqueue(60)
queue2.display()

#Problem 7, 8, 9
print("\nProblem 7, 8, 9")

class PriorityQueue:
    def __init__(self):
        self.queue = []

    def enqueue(self, item, priority):

        self.queue.append((priority, item))
        self.queue.sort(key=lambda x: x[0])

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)
        else:
            return "Priority Queue is empty."

    def display(self):
        if self.is_empty():
            return "Priority Queue is empty."
```

```
        return [(priority, item) for priority, item in self.queue]

    def sorted_display(self):
        return self.display()

    def is_empty(self):
        return len(self.queue) == 0




queue3 = PriorityQueue()



print("Enqueuing elements:")
queue3.enqueue("Task A", 3)
queue3.enqueue("Task B", 1)
queue3.enqueue("Task C", 2)
queue3.enqueue("Task D", 3)
queue3.enqueue("Task E", 1)
print("Current Priority Queue:", queue3.display())



print("\nDisplaying sorted priority queue:")
print(queue3.sorted_display())



print("\nDequeuing the highest-priority element:")
dequeued = queue3.dequeue()
```

```
print(f"Dequeued element: {dequeued}")
print("Priority Queue after dequeue:", queue3.display())


print("\nTesting with duplicate priorities:")
queue3.enqueue("Task F", 3)
queue3.enqueue("Task G", 1)
print("Priority Queue after adding duplicates:", queue3.display())


print("\nDequeuing all elements:")
while not queue3.is_empty():
    print(f"Dequeued element: {queue3.dequeue()}")
    print("Priority Queue state:", queue3.display())


print("\nChecking if the queue is empty:")
print(f"Is the queue empty? {queue3.is_empty()}")
```