```
# Queue Implementation using Array
class ArrayQueue:
    def __init__(self, capacity):
        self.capacity = capacity
        self.queue = []

    def enqueue(self, item):
        if len(self.queue) < self.capacity:
            self.queue.append(item)
        else:
            print("Array Queue is full!")

    def dequeue(self):
        if self.queue:
            return self.queue.pop(0)
        else:
            return "Array Queue is empty!"

# Queue Implementation using Linked List
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedListQueue:
    def __init__(self):
        self.front = self.rear = None
```

```
    def enqueue(self, item):
        new_node = Node(item)
        if self.rear:
            self.rear.next = new_node
        self.rear = new_node
        if not self.front:
            self.front = new_node

    def dequeue(self):
        if self.front:
            temp = self.front.data
            self.front = self.front.next
            if not self.front:
                self.rear = None
            return temp
        else:
            return "Linked List Queue is empty!"

# Queue Implementation using Two Stacks
class StackQueue:
    def __init__(self):
        self.stack1 = []
        self.stack2 = []

    def enqueue(self, item):
        self.stack1.append(item)

    def dequeue(self):
```

```
        if not self.stack2:
            while self.stack1:
                self.stack2.append(self.stack1.pop())
        return self.stack2.pop() if self.stack2 else "Stack Queue is empty!"

# Priority Queue using Min-Heap (Tree concept)
import heapq

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

    def enqueue(self, item):
        heapq.heappush(self.heap, item)

    def dequeue(self):
        if self.heap:
            return heapq.heappop(self.heap)
        else:
            return "Priority Queue is empty!"

# Testing all implementations
print("Array Queue:")
aq = ArrayQueue(3)
aq.enqueue(1)
aq.enqueue(2)
aq.enqueue(3)
aq.enqueue(4)  # Should not be added
```

```
print([aq.dequeue() for _ in range(4)])

print("\nLinked List Queue:")
llq = LinkedListQueue()
llq.enqueue(10)
llq.enqueue(20)
llq.enqueue(30)
print([llq.dequeue() for _ in range(4)])

print("\nStack Queue:")
sq = StackQueue()
sq.enqueue(100)
sq.enqueue(200)
sq.enqueue(300)
print([sq.dequeue() for _ in range(4)])

print("\nPriority Queue:")
pq = PriorityQueue()
pq.enqueue(5)
pq.enqueue(1)
pq.enqueue(3)
pq.enqueue(2)
print([pq.dequeue() for _ in range(5)])
```