Priority Queue using Linked List - Detailed Dry Run

Step 1: Initialization

• Start = null (empty queue)
No nodes in memory.

Step 2: Insert (10,1)

• Queue empty, so new node becomes start.
• start -> (10, priority=1)

Step 3: Insert (20,2)

• New node (20,2) has higher priority than start (10,1).
• Insert at front: new node points to old start.
• start -> (20,2) -> (10,1)

Step 4: Insert (30,3)

• New node (30,3) has higher priority than current start (20,2).
• Insert at front again.
• start -> (30,3) -> (20,2) -> (10,1)

Step 5: peekHighest()

• Return start.item = 30
• Queue remains unchanged.

Step 6: deleteHighestPriority()

• Highest priority is always at start.
• Delete (30,3). Update start to point to next node.
• start -> (20,2) -> (10,1)

Step 7: Display()

• Traverse linked list from start.
• Output: [(20, priority=2) (10, priority=1)]

■ Key Observations

1. Insert takes O(n) in worst case (need to traverse to correct position).
2. Peek and Delete take O(1), since highest priority always at front.
3. This implementation maintains descending order of priority.