**C-DAC Mumbai Date 01/10/2024**

**Subject: Algorithm and Data Structure**

**Assignment 3**

**Solve the assignment with following thing to be added in each question.**

-Program

-Flow chart

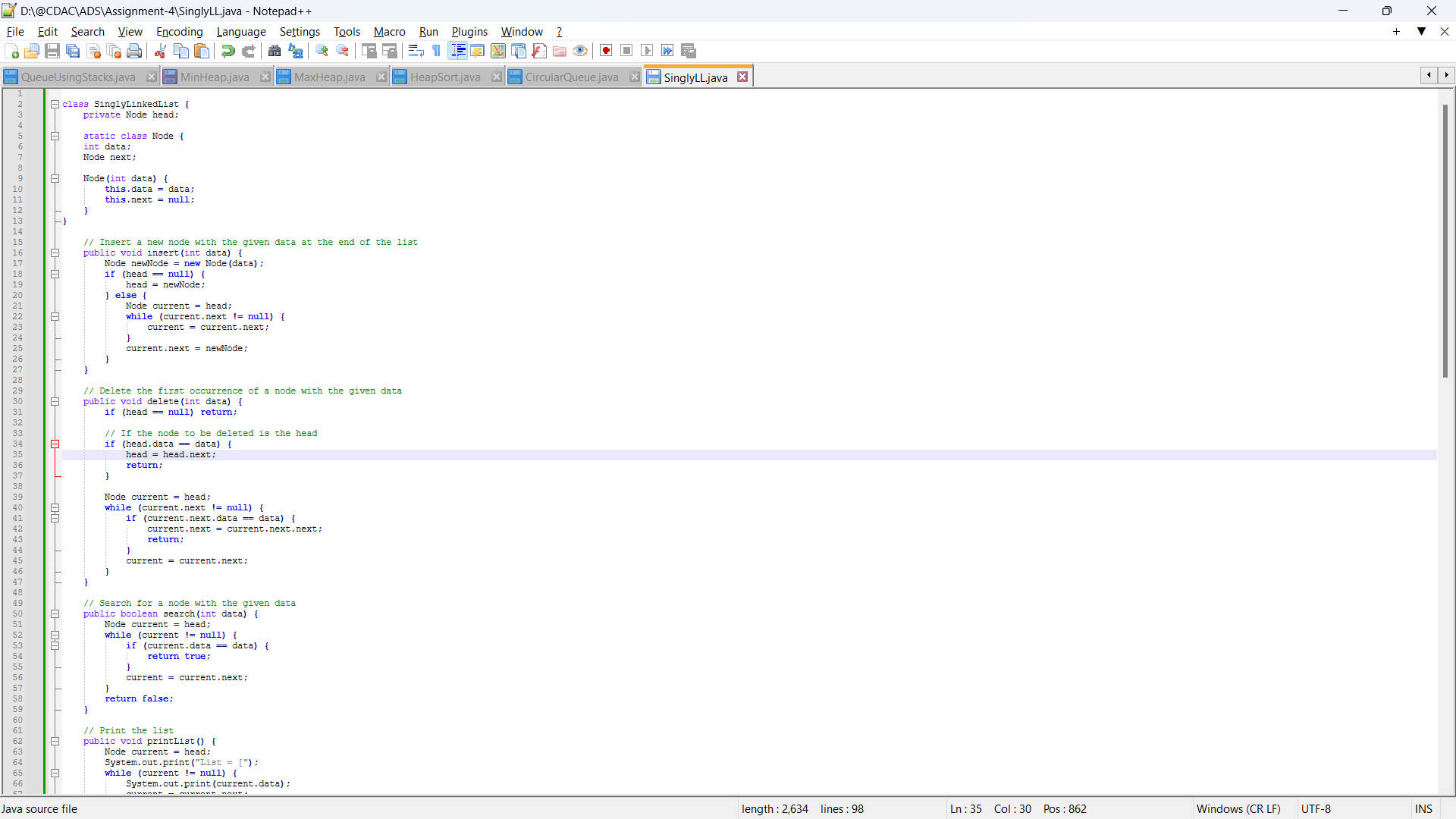
-Explanation

-Output

-Time and Space complexity

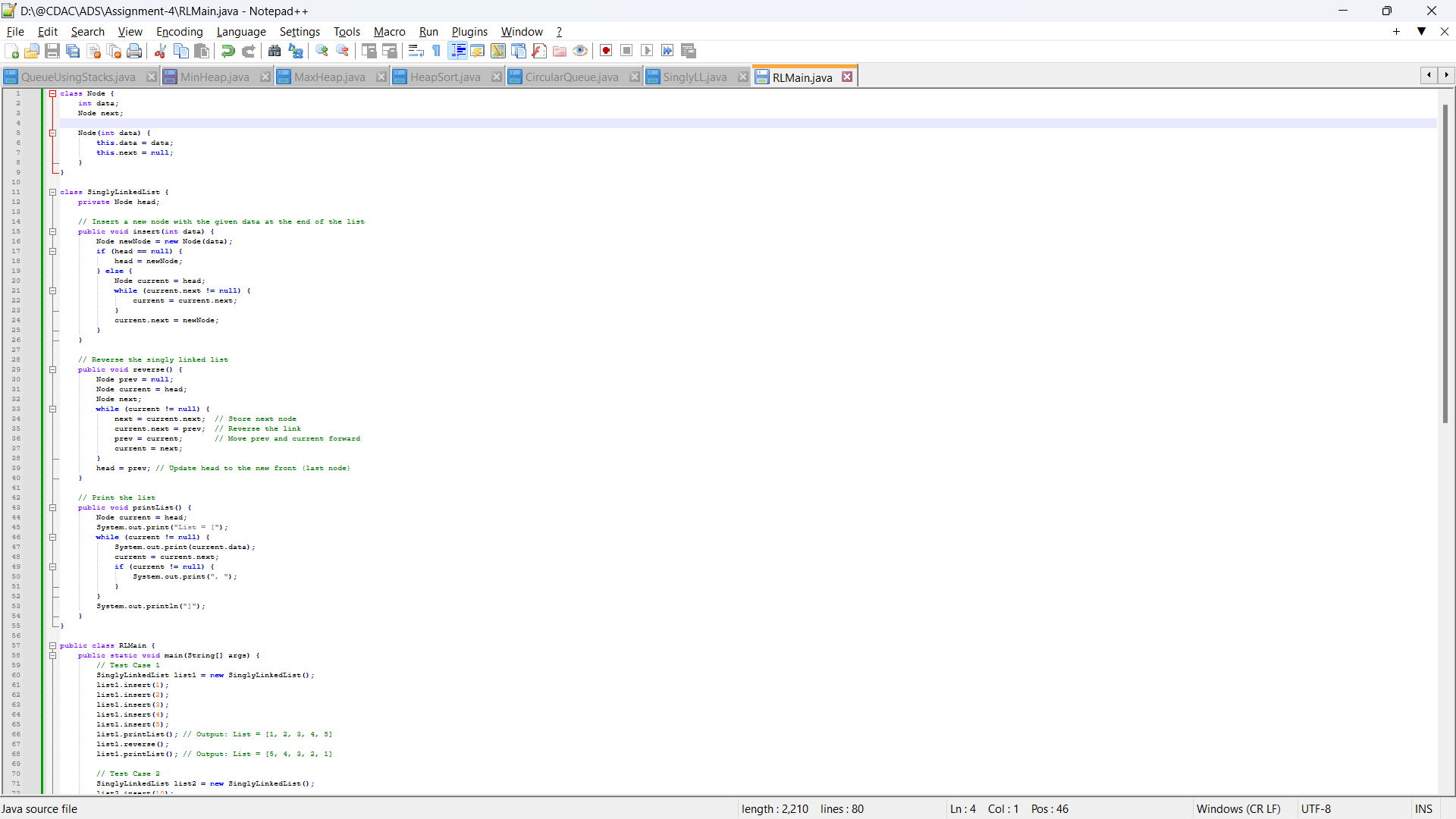
Submission Date: 3/10/2024

**1. Implement a singly linked list with basic operations: insert, delete, search.**

* **Test Case 1**:  
  Input: Insert 3 → Insert 7 → Insert 5 → Delete 7 → Search 5  
  Output: List = [3, 5], Found = True
* **Test Case 2**:  
  Input: Insert 9 → Insert 4 → Delete 4 → Search 10  
  Output: List = [9], Found = False
* 

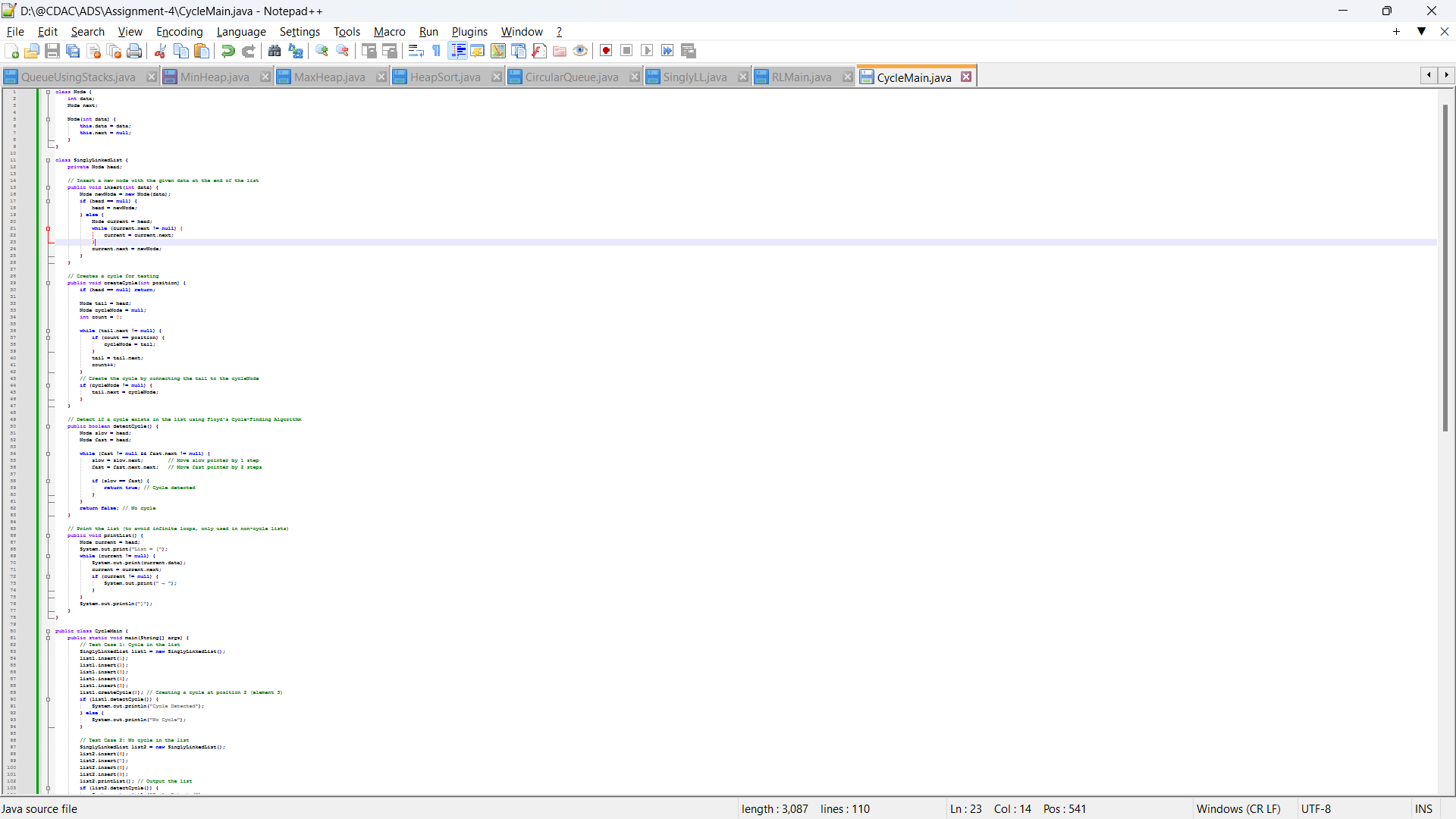
**2. Reverse a singly linked list.**

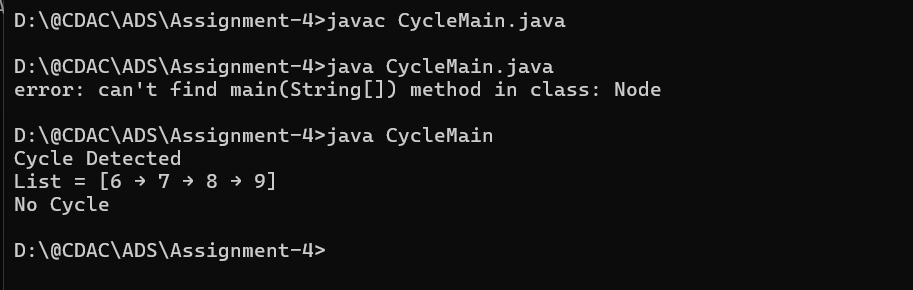
* **Test Case 1**:  
  Input: List = [1, 2, 3, 4, 5]  
  Output: List = [5, 4, 3, 2, 1]
* **Test Case 2**:  
  Input: List = [10, 20, 30]  
  Output: List = [30, 20, 10]



**3. Detect a cycle in a linked list.**

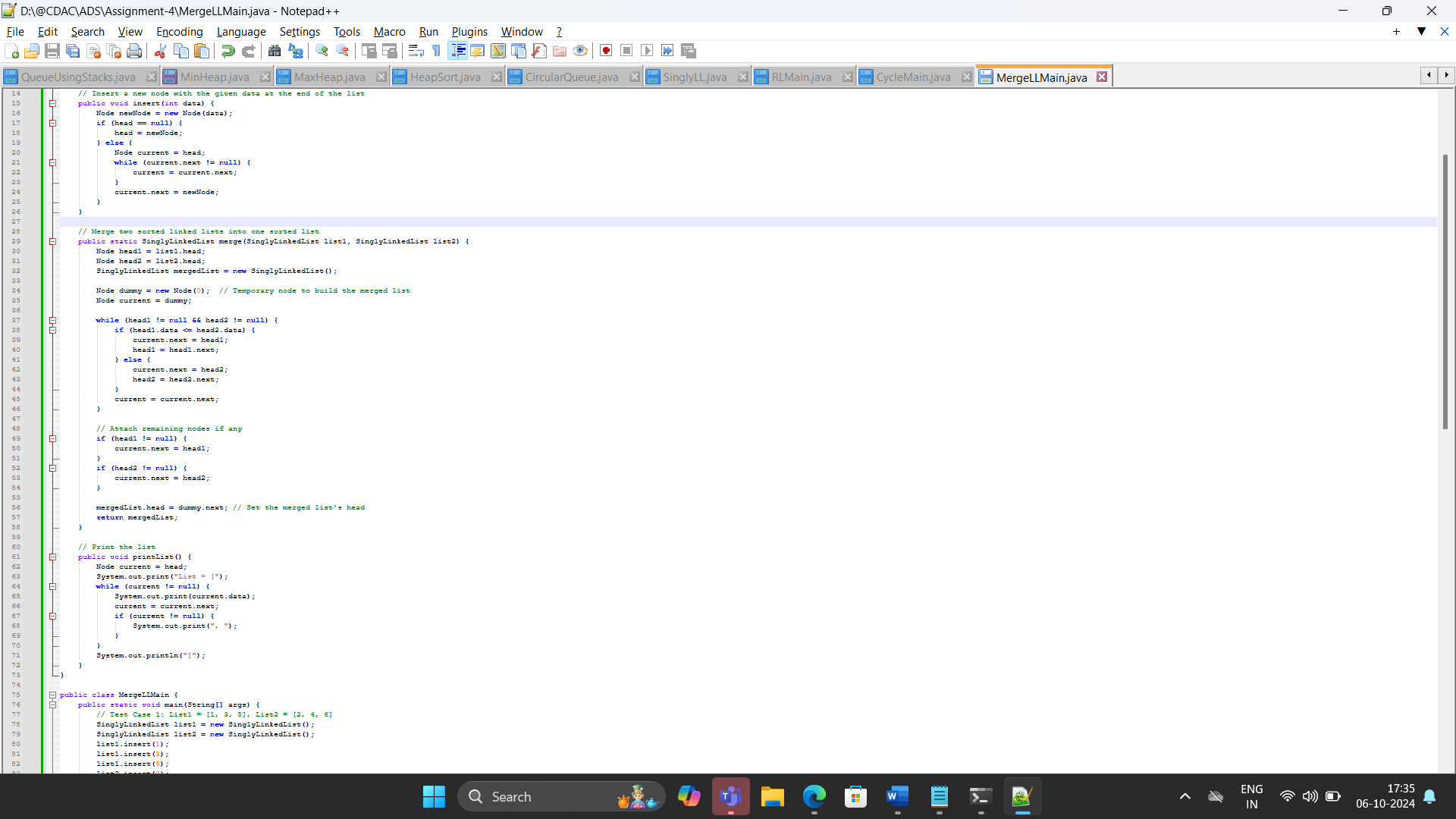
* **Test Case 1**:  
  Input: List = [1 → 2 → 3 → 4 → 5 → 3 (cycle)]  
  Output: Cycle Detected
* **Test Case 2**:  
  Input: List = [6 → 7 → 8 → 9]  
  Output: No Cycle

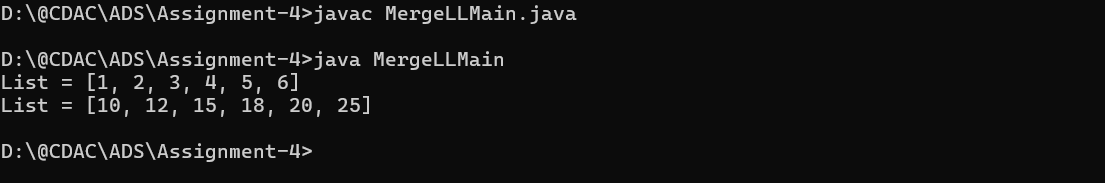




**4. Merge two sorted linked lists.**

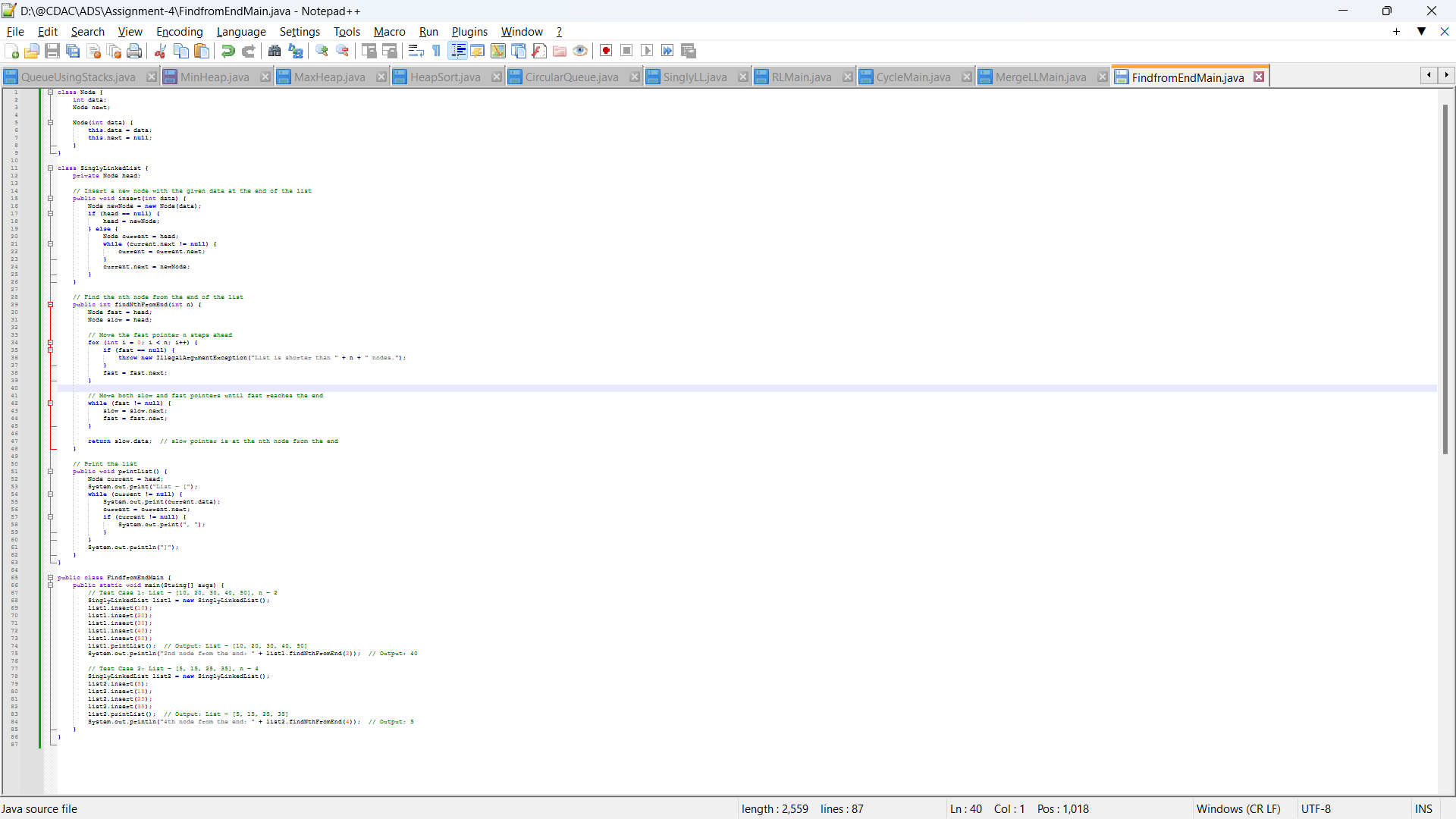
* **Test Case 1**:  
  Input: List1 = [1, 3, 5], List2 = [2, 4, 6]  
  Output: Merged List = [1, 2, 3, 4, 5, 6]
* **Test Case 2**:  
  Input: List1 = [10, 15, 20], List2 = [12, 18, 25]  
  Output: Merged List = [10, 12, 15, 18, 20, 25]

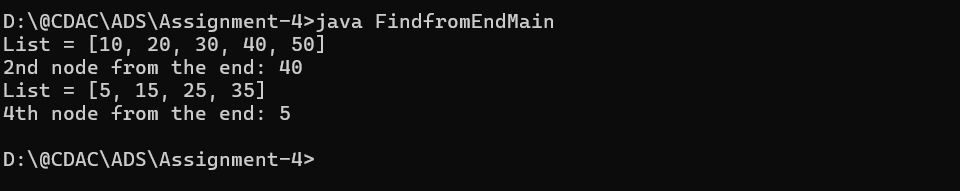




**5. Find the nth node from the end of a linked list.**

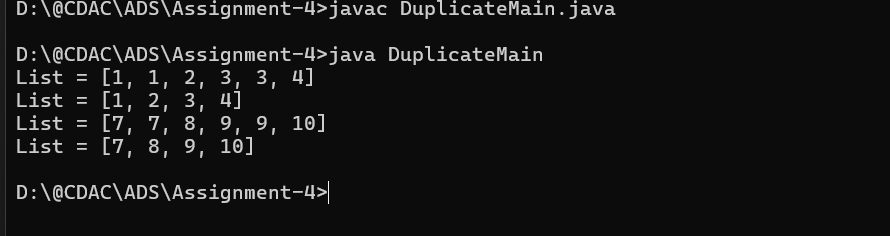
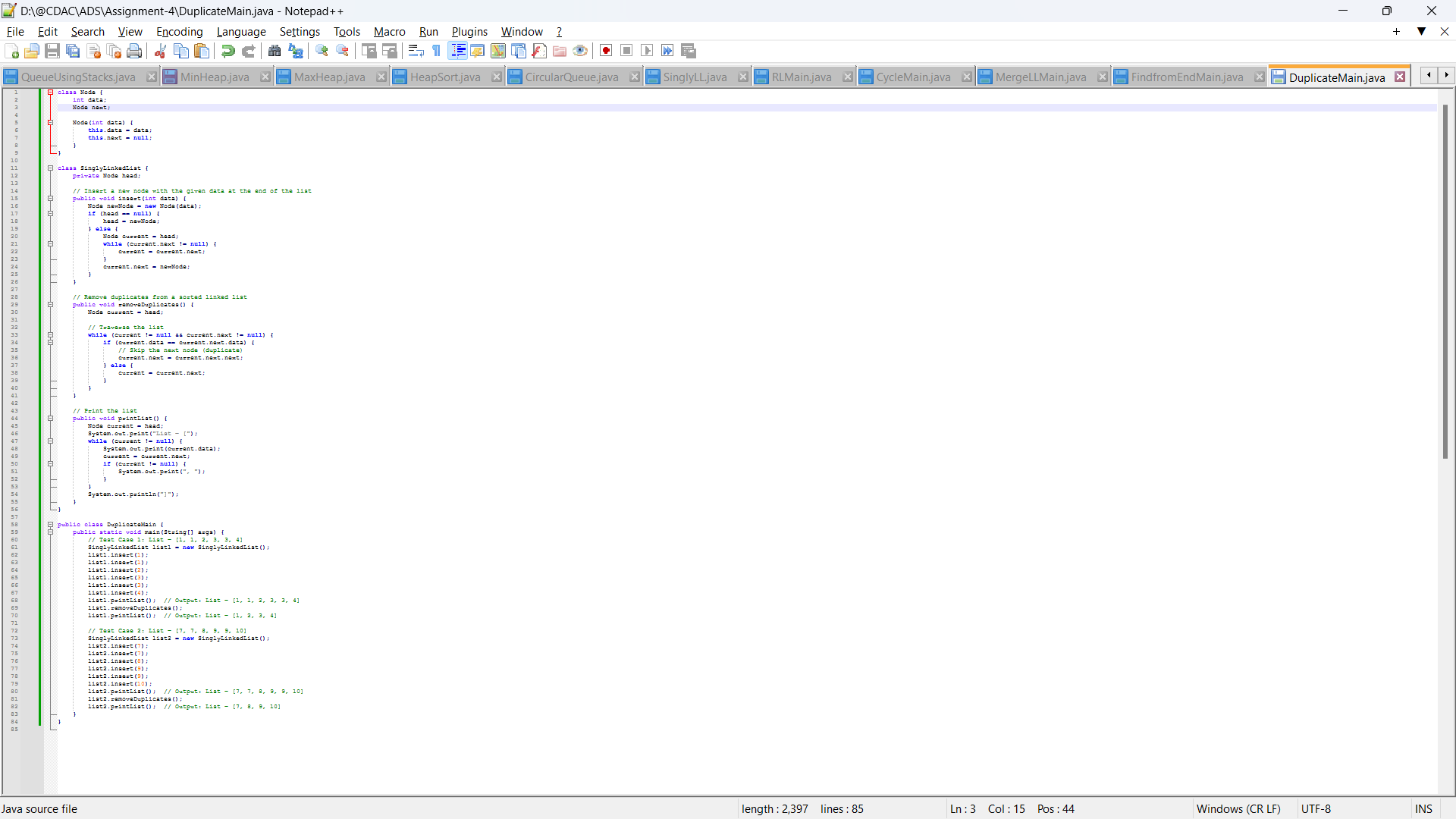
* **Test Case 1**:  
  Input: List = [10, 20, 30, 40, 50], n = 2  
  Output: 40
* **Test Case 2**:  
  Input: List = [5, 15, 25, 35], n = 4  
  Output: 5





**6. Remove duplicates from a sorted linked list.**

* **Test Case 1**:  
  Input: List = [1, 1, 2, 3, 3, 4]  
  Output: List = [1, 2, 3, 4]
* **Test Case 2**:  
  Input: List = [7, 7, 8, 9, 9, 10]  
  Output: List = [7, 8, 9, 10]



**7. Implement a doubly linked list with insert, delete, and traverse operations.**

* **Test Case 1**:  
  Input: Insert 10 → Insert 20 → Insert 30 → Delete 20  
  Output: List = [10, 30]
* **Test Case 2**:  
  Input: Insert 1 → Insert 2 → Insert 3 → Delete 1  
  Output: List = [2, 3]

**8. Reverse a doubly linked list.**

* **Test Case 1**:  
  Input: List = [5, 10, 15, 20]  
  Output: List = [20, 15, 10, 5]
* **Test Case 2**:  
  Input: List = [4, 8, 12]  
  Output: List = [12, 8, 4]

**9. Add two numbers represented by linked lists.**

* **Test Case 1**:  
  Input: List1 = [2 → 4 → 3], List2 = [5 → 6 → 4] (243 + 465)  
  Output: Sum List = [7 → 0 → 8]
* **Test Case 2**:  
  Input: List1 = [9 → 9 → 9], List2 = [1] (999 + 1)  
  Output: Sum List = [0 → 0 → 0 → 1]

**10. Rotate a linked list by k places.**

* **Test Case 1**:  
  Input: List = [10, 20, 30, 40, 50], k = 2  
  Output: List = [30, 40, 50, 10, 20]
* **Test Case 2**:  
  Input: List = [5, 10, 15, 20], k = 3  
  Output: List = [20, 5, 10, 15]

**11. Flatten a multilevel doubly linked list.**

* **Test Case 1**:  
  Input: List = [1 → 2 → 3, 3 → 7 → 8, 8 → 10 → 12]  
  Output: Flattened List = [1 → 2 → 3 → 7 → 8 → 10 → 12]
* **Test Case 2**:  
  Input: List = [1 → 2 → 3, 2 → 5 → 6, 6 → 7 → 9]  
  Output: Flattened List = [1 → 2 → 5 → 6 → 7 → 9 → 3]

**12. Split a circular linked list into two halves.**

* **Test Case 1**:  
  Input: Circular List = [1 → 2 → 3 → 4 → 5 → 6 → (back to 1)]  
  Output: List1 = [1 → 2 → 3], List2 = [4 → 5 → 6]
* **Test Case 2**:  
  Input: Circular List = [10 → 20 → 30 → 40 → (back to 10)]  
  Output: List1 = [10 → 20], List2 = [30 → 40]

**13. Insert a node in a sorted circular linked list.**

* **Test Case 1**:  
  Input: Circular List = [10 → 20 → 30 → 40 → (back to 10)], Insert 25  
  Output: Circular List = [10 → 20 → 25 → 30 → 40 → (back to 10)]
* **Test Case 2**:  
  Input: Circular List = [5 → 15 → 25 → (back to 5)], Insert 10  
  Output: Circular List = [5 → 10 → 15 → 25 → (back to 5)]

**14. Check if two linked lists intersect, and find the intersection point if they do.**

* **Test Case 1**:  
  Input: List1 = [1 → 2 → 3 → 4 → 5], List2 = [6 → 7 → 4 → 5]  
  Output: Intersection Point = 4
* **Test Case 2**:  
  Input: List1 = [10 → 20 → 30 → 40], List2 = [15 → 25 → 35]  
  Output: No Intersection

**15. Find the middle element of a linked list in one pass.**

* **Test Case 1**:  
  Input: List = [1, 2, 3, 4, 5]  
  Output: Middle = 3
* **Test Case 2**:  
  Input: List = [11, 22, 33, 44, 55, 66]  
  Output: Middle = 44