**📝 Infinite Champions Programme – Day 10 (Assignment Sheet)**

**📌 Instructions  
• Deadline: Submit your solutions by 17th October, 2025, EOD.  
• Platform: Test your solutions on LeetCode  
• Collaboration: Discussing concepts is encouraged, but all code must be your own.**

1. [**Min Stack (LeetCode #155)**](https://leetcode.com/problems/min-stack/)  
   • **Problem:** Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.  
   • **Objective:** Use an auxiliary stack to track the current minimum after each operation.  
   • **YouTube Solution (Java):** [Min Stack – Java Solution](https://www.youtube.com/watch?v=8UegNFCUQks)
2. [**Implement Stack using Queues (LeetCode #225)**](https://leetcode.com/problems/implement-stack-using-queues/)  
   • **Problem:** Implement a stack using only two queues.  
   • **Objective:** Use one queue to simulate LIFO behavior by rotating elements after each push.  
   • **YouTube Solution (Java):** [Implement Stack using Queues – Java Solution](https://www.youtube.com/watch?v=jDZQKzEtbYQ)
3. [**Remove K Digits (LeetCode #402)**](https://leetcode.com/problems/remove-k-digits/)  
   • **Problem:** Given a non-negative integer represented as a string, remove *k* digits to make the smallest possible number.  
   • **Objective:** Use a stack to maintain a monotonically increasing sequence of digits.  
   • **YouTube Solution (Java):** [Remove K Digits – Java Solution](https://www.youtube.com/watch?v=3QJzHqNAEXs)
4. [**Asteroid Collision (LeetCode #735)**](https://leetcode.com/problems/asteroid-collision/)  
   • **Problem:** Given an array representing moving asteroids, determine the state after all collisions.  
   • **Objective:** Use a stack to simulate the collision process based on direction and size.  
   • **YouTube Solution (Java):** [Asteroid Collision – Java Solution](https://www.youtube.com/watch?v=LN7KjRszjk4)
5. [**Basic Calculator II (LeetCode #227)**](https://leetcode.com/problems/basic-calculator-ii/)  
   • **Problem:** Evaluate a simple mathematical expression string containing +, -, \*, and /.  
   • **Objective:** Use a stack to store intermediate results and handle operator precedence.  
   • **YouTube Solution (Java):** [Basic Calculator II – Java Solution](https://www.youtube.com/watch?v=UghXo5V3HAc)
6. [**Car Fleet (LeetCode #853)**](https://leetcode.com/problems/car-fleet/)  
   • **Problem:** Given positions and speeds of cars, find how many car fleets will arrive at the destination.  
   • **Objective:** Use a stack to track the time taken by each car and merge fleets moving at compatible speeds.  
   • **YouTube Solution (Java):** [Car Fleet – Java Solution](https://www.youtube.com/watch?v=Pr6T-3yB9RM)
7. [**Decode String (LeetCode #394)**](https://leetcode.com/problems/decode-string/)  
   • **Problem:** Decode an encoded string like 3[a2[c]] → accaccacc.  
   • **Objective:** Use two stacks to handle nested multipliers and character sequences.  
   • **YouTube Solution (Java):** [Decode String – Java Solution](https://www.youtube.com/watch?v=qB0zZpBJlh8)
8. [**Next Greater Element I (LeetCode #496)**](https://leetcode.com/problems/next-greater-element-i/)  
   • **Problem:** For each element in one array, find the next greater element in another array.  
   • **Objective:** Use a monotonic stack to find the next greater elements efficiently.  
   • **YouTube Solution (Java):** [Next Greater Element I – Java Solution](https://www.youtube.com/watch?v=68a1Dc_qVq4)
9. [**Remove All Adjacent Duplicates in String II (LeetCode #1209)**](https://leetcode.com/problems/remove-all-adjacent-duplicates-in-string-ii/)  
   • **Problem:** Remove all adjacent duplicates in a string when a character appears *k* times consecutively.  
   • **Objective:** Use a stack to track characters and their counts to remove duplicates efficiently.  
   • **YouTube Solution (Java):** [Remove All Adjacent Duplicates in String II – Java Solution](https://www.youtube.com/watch?v=QkD5a3MD1ho)
10. [**Maximal Rectangle (LeetCode #85)**](https://leetcode.com/problems/maximal-rectangle/)  
    • **Problem:** Find the largest rectangle containing only 1’s in a binary matrix.  
    • **Objective:** Use a stack to compute the largest rectangle in each row’s histogram.  
    • **YouTube Solution (Java):** [Maximal Rectangle – Java Solution](https://www.youtube.com/watch?v=vsx-Ku2M3rE)

**📚 Submission Checklist  
• Time and space complexity analysis for each solution.  
• Test cases demonstrating the correctness of your solutions.**