Exam-Style Problem Sheet: Stack and Queue Based Logical Problems

---

Problem 1: Merge Overlapping Intervals (Stack Only)

You are given a stack containing pairs representing intervals sorted by start time from **bottom to top**. Merge all overlapping intervals. After merging, print the start and end time of each non-overlapping interval in **descending order of start time**.

Constraints: - You may only use Stack operations. - No arrays, recursion, or external sorting allowed.

Problem 2: Job Scheduling with Deadlines (Stack Simulation)

Each element in a stack represents a job in the form of `(job_id, deadline, profit)`, sorted by deadline from bottom to top. Schedule the jobs to maximize total profit such that only one job can be done per time unit.

Input: Stack of jobs where each job is represented as a tuple.

Output: Print the selected job IDs in descending order of time.

Constraints: - Use only stacks to store jobs and simulate time.

Problem 3: Maximum in Sliding Window (Queue Only)

You are given a queue of integers and a window size `k`. Find and print the **maximum** in each sliding window of size `k`.

Constraints: - Use only Queue operations. - No arrays, deques, or recursion allowed.

Input Example: Queue: 2, 1, 3, 4, 6, 3, 8, 9, 10, 12, 56
Window size: 4

Output: 4 6 6 8 9 10 12 56

Problem 4: Detect Redundant Parentheses (Stack Based)

Given a stream of characters representing a mathematical expression using a Queue, detect whether it contains **redundant parentheses**.

Example: Input: ((a+b)) → Output: Yes
Input: (a+(b)/c) → Output: No

Constraints: - Use only a Stack and Queue. - No string manipulation functions allowed.

Problem 5: Task Conflict Detection (Queue Based)

Given a queue of tasks, where each task is represented as a pair `(start, end)` and sorted by start time, detect if any two tasks **overlap**.

Input: Queue of tasks like (1, 3), (2, 5), (6, 8)

Output: Print YES if any overlap exists, otherwise NO.

Constraints: - Use only queues for implementation.

Problem 6: Decode Encoded Strings (Stack Simulation)

Given an encoded string in the format like `3[a2[b]]`, decode it and print the final result.

Example: Input: 3[a2[b]] → Output: abbabbabb

Constraints: - Only use stacks. - No use of arrays, recursion or built-in parsers.

Problem 7: Sorting a Queue Using Stack

Sort a queue of integers in **ascending order** using only **one queue and one stack**.

Constraints: - No additional data structures (like arrays) or recursion allowed. - Final queue must be sorted with smallest element at front.

Instructions: - Use only stack and queue operations as defined. - Show all intermediate steps if asked. - Time complexity analysis is not required unless explicitly mentioned.