🧱 A. Basic Stack Tasks

1. Create a basic stack using array

Steps:

- Create an array.
- Use push() to add.
- Use pop() to remove.
- Use length and peek.

2. Create Stack class with push(), pop(), peek()

Steps:

- Use class Stack.
- Add constructor with this.items = [].
- Define all 3 methods with proper return.

3. Check if the stack is empty

Steps:

- Add method isEmpty() returning this.items.length === 0.

4. Print all elements using printStack()

Steps:

- Use join(" ") to print items.

5. Reverse a string using stack

Steps:

- Loop through string characters and push() them.
- Pop elements back to get reversed string.

6. Find palindrome using stack

Steps:

- Use a stack to reverse a string.
- Compare original with reversed.

7. Match opening/closing brackets using stack

Steps:

- Push (or { or [into stack.
- Pop when matching closing bracket found.
- Check for underflow/mismatch.

8. Stack underflow condition

Steps:

- Return "Underflow" if pop() or peek() on empty stack.

9. Convert decimal to binary using stack

Steps:

- Divide number by 2, push remainders.
- Pop and join to get binary.

---

10. Use stack to implement undo-redo

Steps:

- Two stacks: undo and redo.
- On change, push to undo.
- On undo, pop from undo to redo.

📤 B. Basic Queue Tasks

---

11. Create basic queue using array

Steps:

- Use push() for enqueue, shift() for dequeue.

---

12. Create Queue class with enqueue(), dequeue(), peek()

Steps:

- Define class Queue with items = [].

---

13. Print queue elements

Steps:

- Implement printQueue() using join(" ").

---

14. Check if the queue is empty

Steps:

- this.items.length === 0.

---

15. Validate underflow in queue

Steps:

- Return "Underflow" if dequeue() called on empty.

---

16. Simulate a printer queue

Steps:

- Add document names using enqueue().
- Process using dequeue() with setTimeout() to simulate time delay.

---

17. Queue-based customer service

Steps:

- Add customers with enqueue().
- Serve with dequeue() every X seconds.

---

18. CPU task queue simulation

Steps:

- Use array of tasks.
- Add timestamps.
- Simulate completion and scheduling.

---

19. Generate Fibonacci series using queue

Steps:

- Keep last two numbers in queue.
- Enqueue next value, dequeue oldest.

---

20. Implement queue for real-time message buffering

Steps:

- Add messages using enqueue().
- Print/flush every 2 seconds using interval.

🔄 C. Combined Stack & Queue Challenges

---

21. Use stack to reverse elements of a queue

Steps:

- Dequeue all elements and push to stack.
- Pop from stack and enqueue back.

---

22. Evaluate postfix expression using stack

Steps:

- Read each character.
- Push number, pop for operators, evaluate.

---

23. Implement queue using two stacks

Steps:

- Use stack1 for enqueue.
- On dequeue, pop all to stack2, pop top.

---

24. Implement stack using two queues

Steps:

- Push to queue1.
- On pop(), transfer all except last to queue2, then swap queues.

---

25. Sort a stack using another stack

Steps:

- Use temporary stack.
- Insert in sorted order by comparing top values.

---

26. Browser back-forward navigation

Steps:

- Use 2 stacks: backStack, forwardStack.
- On visit(), push to backStack.
- On back(), pop to forwardStack.

⭕ D. Bonus: Circular Queue Tasks

27. Create circular queue structure

Steps:

- Use array of fixed size.
- Track front, rear, wrap using modulo %.

28. Circular queue overflow and underflow

Steps:

- enqueue() should not go beyond capacity.
- dequeue() should handle wrap.

29. Use circular queue for music playlist

Steps:

- Add songs.
- Automatically loop back after last.

30. Build real-time event loop using queue

Steps:

- Add events dynamically.
- Process them FIFO.
- Handle asynchronous timing.

📘 E. Debug & Fix Tasks (Optional)

31. Fix a buggy stack class

Task:

- Provide class with missing return, wrong array use.
- Students fix bugs (e.g., this.items = item instead of []).

32. Validate mixed operations on both Stack & Queue

Task:

- Perform alternate stack and queue operations.
- Maintain expected state tracking.

33. Implement history tracking for input values

Task:

- Add stack for inputs.
- Undo inputs via pop().
- Redo with a second stack.

✅ Summary Table of Learning Objectives

Task Range	Level	Concepts Covered
1–4	Beginner	Stack basics with array and class
5–10	Intermediate	Real-life stack use cases
11–15	Beginner	Queue basics, enqueue, dequeue, validation
16–20	Intermediate	Queue in real applications
21–26	Advanced	Inter-conversion, navigation, evaluation
27–30	Advanced	Circular queue and continuous data handling
31–33	Debugging	Error fixing and logic testing