**Beginner Level:**

1. Implement a stack using an array
2. Implement a stack using a linked list
3. Implement stack operations: push, pop, peek, and isEmpty
4. Reverse a string using a stack
5. Check for balanced parentheses in an expression
6. Evaluate a postfix expression
7. Convert an infix expression to a postfix expression
8. Convert an infix expression to a prefix expression
9. Evaluate a prefix expression
10. Sort a stack using another stack
11. Find the minimum element in a stack in O(1) time
12. Implement two stacks using a single array
13. Design a stack that supports getMin() in O(1) time
14. Implement a stack using queues
15. Implement a stack that supports middle element retrieval
16. Find the next greater element for each element in an array
17. Find the previous greater element for each element in an array
18. Find the next smaller element for each element in an array
19. Find the previous smaller element for each element in an array
20. Check if a stack permutation is valid

**Medium Level:**

1. Implement a stack that supports increment operations
2. Design a stack with operations on middle element
3. Check for duplicate parentheses in an expression
4. Implement the Stock Span Problem
5. Find the maximum rectangular area in a histogram
6. Design a circular buffer using a stack
7. Implement a special stack with getMin() and getMax()
8. Remove adjacent duplicates from a string using a stack
9. Decode a string encoded with numbers and brackets (e.g., "3[a]2[bc]")
10. Find the longest valid parentheses substring
11. Implement a stack-based version of the Tower of Hanoi
12. Find the celebrity in a group of people
13. Implement a stack-based palindrome checker
14. Merge overlapping intervals using a stack
15. Check if an expression has redundant brackets
16. Implement a basic calculator using a stack
17. Implement a stack-based expression evaluator for mixed operators
18. Simplify a Unix-style file path using a stack
19. Implement a stack to sort elements in place
20. Use a stack to reverse individual words in a string

**Advanced Level:**

1. Implement a multi-level stack
2. Implement a dynamic stack that grows and shrinks as needed
3. Implement a stack with a custom memory allocator
4. Use a stack to solve the water trapped between bars problem
5. Implement a stack-based scheduling algorithm
6. Implement a non-recursive postorder traversal of a binary tree
7. Design a stack that supports undo and redo operations
8. Use a stack to solve the max sliding window problem
9. Implement a stack-based approach to find the longest common subsequence
10. Use a stack to convert a binary search tree to a doubly linked list