50 Stack Practice Problems

1. Push and Pop Elements

Input:
Operations: PUSH 10, PUSH 20, POP, PUSH 30

Output:
Stack: [10, 30]

2. Check if Stack is Empty

Input:
Operations: POP, PUSH 5, POP

Output:
Is Empty: true

3. Peek the Top Element

Input:
PUSH 5, PUSH 10, PEEK

Output:
Top Element: 10

4. Print All Stack Elements

Input:
PUSH 1, PUSH 2, PUSH 3

Output:
Stack: [1, 2, 3]

5. Find Size of the Stack

Input:
PUSH 10, PUSH 20, PUSH 30

Output:
Size: 3

6. Reverse a Stack

Input:
Stack: [1, 2, 3, 4, 5]

Output:
Reversed: [5, 4, 3, 2, 1]

7. Insert at Bottom of Stack

Input:
Stack: [2, 3, 4], Insert: 1

Output:
[1, 2, 3, 4]

8. Sort a Stack using Another Stack

Input:
Stack: [3, 1, 4, 2]

Output:
Sorted: [1, 2, 3, 4]

9. Check for Balanced Parentheses

Input:
Expression: (a + b) * (c + d)

Output:
Balanced: true

10. Convert Decimal to Binary using Stack

Input:
Number: 13

Output:
Binary: 1101

11. Evaluate Postfix Expression

Input:
Postfix: 2 3 * 5 +

Output:
Result: 11

12. Convert Infix to Postfix

Input:
Infix: A + B * C

Output:
Postfix: A B C * +

13. Convert Infix to Prefix

Input:
Infix: (A+B)*C

Output:
Prefix: *+ABC

14. Delete Middle Element from Stack

Input:
Stack: [1, 2, 3, 4, 5]

Output:
After Deletion: [1, 2, 4, 5]

15. Check Palindrome Using Stack

Input:
String: racecar

Output:
Palindrome: true

16. Next Greater Element

Input:
Array: [4, 5, 2, 25]

Output:
Next Greater: [5, 25, 25, -1]

17. Next Smaller Element

Input:
Array: [4, 5, 2, 10, 8]

Output:
Next Smaller: [2, 2, -1, 8, -1]

18. Previous Greater Element

Input:
Array: [10, 4, 2, 20, 40, 12, 30]

Output:
Prev Greater: [-1, 10, 4, -1, -1, 40, 40]

19. Previous Smaller Element

Input:
Array: [1, 3, 0, 2, 5]

Output:
Prev Smaller: [-1, 1, -1, 0, 2]

20. Stock Span Problem

Input:
Prices: [100, 80, 60, 70, 60, 75, 85]

Output:
Span: [1, 1, 1, 2, 1, 4, 6]

21. Largest Rectangle in Histogram

Input:
Heights: [6, 2, 5, 4, 5, 1, 6]

Output:
Max Area: 12

22. Celebrity Problem

Input:
Matrix:
0 1 1
0 0 1
0 0 0

Output:
Celebrity ID: 2

23. Sort Stack Recursively

Input:
Stack: [34, 3, 31, 98, 92, 23]

Output:
Sorted: [3, 23, 31, 34, 92, 98]

24. Insert Element at Given Index in Stack

Input:
Stack: [1, 2, 3, 4], Index: 2, Element: 99

Output:
[1, 2, 99, 3, 4]

25. Check for Redundant Brackets

Input:
Expression: ((a+b))

Output:
Redundant: true

26. Remove Adjacent Duplicates in Stack

Input:
String: azxxzy

Output:
Result: ay

27. Find Minimum Element in Constant Time

Input:
PUSH 5, PUSH 3, PUSH 7, MIN

Output:
Minimum: 3

28. Design Stack Supporting GetMin

Input:
PUSH 2, PUSH 6, PUSH 1, GETMIN, POP, GETMIN

Output:
Min1: 1, Min2: 2

29. Reverse a String Using Stack

Input:
String: hello

Output:
olleh

30. Check if String is Valid Parenthesis Expression

Input:
String: {[()()]()}

Output:
Valid: true

31. Decode a String with Nested Encoding

Input:
String: 3[a2[c]]

Output:
accaccacc

32. Evaluate Prefix Expression

Input:
Prefix: + 9 * 2 6

Output:
Result: 21

33. Evaluate Infix Expression

Input:
Infix: (1+(4+5+2)-3)+(6+8)

Output:
Result: 23

34. Implement Stack using Queues

Input:
PUSH 1, PUSH 2, POP, TOP

Output:
Top: 1

35. Implement Queue using Stacks

Input:
ENQUEUE 1, ENQUEUE 2, DEQUEUE

Output:
Dequeued: 1

37. K Stacks in a Single Array

Input:
N = 10, K = 3
Operations: push(0, 15), push(1, 45), push(2, 17), pop(1)

Output:
Popped from stack 1: 45

38. Design Stack with Increment Operation

Input:
Stack: [1, 2, 3], Increment: 2 by 100

Output:
[101, 102, 3]

39. Sort a Stack in Descending Order

Input:
Stack: [2, 9, 1, 6]

Output:
Sorted Desc: [9, 6, 2, 1]

40. Sort a Stack Using Merge Sort Logic

Input:
Stack: [5, 1, 4, 2]

Output:
Sorted: [1, 2, 4, 5]

42. Design Special Stack with O(1) Min and Max

Input:
PUSH 5, PUSH 1, GETMIN, GETMAX

Output:
Min: 1, Max: 5

43. Delete All Occurrences of an Element from Stack

Input:
Stack: [1, 2, 3, 2, 4, 2], Delete: 2

Output:
[1, 3, 4]

44. Reverse Stack Without Extra Space (Recursion Only)

Input:
Stack: [1, 2, 3]

Output:
[3, 2, 1]

45. Implement Two Stacks in One Array

Input:
Array size: 10
Stack1: push(1), push(2)
Stack2: push(9), push(8)

Output:
Stack1: [1, 2], Stack2: [9, 8]

48. Evaluate Expression with All Brackets and Priorities

Input:
Infix: [(3+2)*(5-1)]/2

Output:
Result: 10

49. Minimum Number of Bracket Reversals

Input:
Expression: "}}}{{{"

Output:
Reversals needed: 3

50. Minimum Add to Make Parentheses Valid

Input:
String: "((("

Output:
Minimum add: 3