Stacks

1. Implement Stack from Scratch

2. Stacks using arrays,LL(pointers)

3. Implement 2 stack in an array

4. find the middle element of a stack.Design a stack with operations on middle element

5. Implement "N" stacks in an Array
# How to efficiently implement k stacks in a single array?

# How to create mergable stack?

6. Check the expression has valid or Balanced parenthesis or not.

7. Reverse a String using Stack

8. Design a Stack that supports getMin() in O(1) time and O(1) extra space.

9. Find the next Greater element

# Next Greater Element

# Next Greater Frequency Element

# Number of NGEs to the right

# Maximum product of indexes of next greater on left and right

10. The celebrity Problem

11. Arithmetic Expression evaluation

12. Evaluation of Postfix expression

13. Implement a method to insert an element at its bottom without using any other data structure.

14. Reverse a stack using recursion

# Reverse the lements of stack using only stack operations (push & pop0 in a non recursive manner.
# Given a stack how to reverse the elements of the stack without using any other data-structure. You cannot
use another stack too.
Time Complexity and Space Complexity is wrong, it is O(n) for both cases.
Hint: Use recursion (system stack.) When you go inside the stack pop elements from stack in each
subsequent call until stack is empty. Then push these elements one by one when coming out of the
recursion. The elements will be reversed.

15. Sort a Stack using recursion

# Sort a stack using a temporary stack

# Delete middle element of a stack

# Sorting array using Stacks

# Delete array elements which are smaller than next or become smaller

# Check if a queue can be sorted into another queue using a stack

# Reverse individual words

# Count subarrays where second highest lie before highest

# Check if an array is stack sortable

16. Merge Overlapping Intervals

17. Largest rectangular Area in Histogram

18. Length of the Longest Valid Substring

19. Expression contains redundant bracket or not

20. Implement Stack using Queue

21. Implement Stack using Deque

22. Stack Permutations (Check if an array is stack permutation of other)
# Catalan number & Dynamic programming concept

23. Implement Queue using Stack

24. Balancing of symbols,Blanced parenthesis checker

# Minimum number of bracket reversals needed to make an expression balanced

# Identify and mark unmatched parenthesis in an expression

# Check if two expressions with brackets are same

# Find index of closing bracket for a given opening bracket in an expression

# Check for balanced parentheses in an expression

# Balanced expression with replacement

# Check if a given array can represent Preorder Traversal of Binary Search Tree

# Form minimum number from given sequence

# Find if an expression has duplicate parenthesis or not

# Find maximum difference between nearest left and right smaller elements

# Find next Smaller of next Greater in an array

# Find maximum sum possible equal sum of three stacks

# Count natural numbers whose all permutation are greater than that number

# Delete consecutive same words in a sequence

# Decode a string recursively encoded as count followed by substring

# Bubble sort using two Stacks

# Pattern Occurrences : Stack Implementation Java

25. Infix-to-postfix conversion

26. Implementing function calls (including recursion)

27. Finding of spans (finding spans in stock markets, refer to Problems section)
# STOCK SPAN RANGE PROBLEM

28. Page-visited history in a Web browser [Back Buttons]

29. Undo sequence in a text editor

30. Matching Tags in HTMLand XML

31.Given an array of characters formed with a’s and b’s. The string is marked with special character X which represents the middle of the list (for example: ababa...ababXbabab baaa). Check whether the string is palindrome.

Implement the above scenario using stacks.

32. Show how to implement one queue efficiently using two stacks. Analyze the running time of the queue operations. (QUEUES)

33. Show how to implement one stack efficiently using two queues. Analyze the running time of the stack operations. (QUEUES)

34. How do we implement two stacks using only one array? Our stack routines should not indicate an exception unless every slot in the array is used?

35. 3 stacks in one array: How to implement 3 stacks in one array?

36. Multiple (m) stacks in one array

37. Suppose there are two singly linked lists which intersect at some point and become a single linked list. The head or start pointers of both the lists are known, but the
intersecting node is not known. Also, the number of nodes in each of the lists before they
intersect are unknown and both lists may have a different number. List1 may have n nodes
before it reaches the intersection point and List2 may have m nodes before it reaches the
intersection point where m and n may be m = n,m < n or m > n. Can we find the merging
point using stacks?

38. Finding Spans: Given an array A, the span S[i] of A[i] is the maximum number of consecutive elements A[j] immediately preceding A[i] and such that A[j] < A[i]?
Other way of asking: Given an array A of integers, find the maximum of j – i subjected to
the constraint of A[i] < A[j].(STOCK MARKET PRICE)

39. On a given machine, how do you check whether the stack grows up or down?
# TEST STACK GROWTH

40. Given a stack of integers, how do you check whether each successive pair of numbers in the stack is consecutive or not. The pairs can be increasing or decreasing, and
if the stack has an odd number of elements, the element at the top is left out of a pair. For
example, if the stack of elements are [4, 5, -2, -3, 11, 10, 5, 6, 20], then the output should
be true because each of the pairs (4, 5), (-2, -3), (11, 10), and (5, 6) consists of
consecutive numbers.(QUEUES)

41.Recursively remove all adjacent duplicates: Given a string of characters, recursively remove adjacent duplicate characters from string. The output string should not
have any adjacent duplicates.

Input: careermonk
Output: camonk
Input: mississippi
Output: m

42. Given an array of elements, replace every element with nearest greater element on the right of that element.

43. How to implement a stack which will support following operations in O(1) time complexity?
• Push which adds an element to the top of stack.
• Pop which removes an element from top of stack.
• Find Middle which will return middle element of the stack.
• Delete Middle which will delete the middle element.

44. Construct a time & space efficient C program to implement stack data structure that
performs push and pop operationsusing only a singl equeue data structure and using

only its enqueue and dequeue operations.

45. Implementation of tower of hanoi using stacks.Iterative Tower of Hanoi

# Print next greater number of Q queries

46.Factorial,fibonacci,matrix inversion using stacks

47. N Queens problem using stacks

48. Backtracking,recursive appli pbms

49. Implement popNode function that will return a first node pointer of the stack

50. Implement the pop function that will return the value at the top of the stack

51. Find if given string is a palindrome or not using a stack.

52. Insert at top & bottom operations using stack

53. : Converting Decimal Numbers to Binary Numbers using stack data structure.
Hint: store reminders into the stack and then print the stack

54. Write a palindrome matching function, which ignore characters other than English alphabet and
digits. String "Madam, I'm Adam." should return true.

55. Iterative Postorder Traversal | Set 1 (Using Two Stacks)

56. Iterative Postorder Traversal | Set 2 (Using One Stack)

57. Print ancestors of a given binary tree node without recursion

58. Find maximum depth of nested parenthesis in a string

59. Find maximum of minimum for every window size in a given array

60. Length of the longest valid substring

Iterative method to find ancestors of a given binary tree
Stack Permutations (Check if an array is stack permutation of other)
Tracking current Maximum Element in a Stack
Check mirror in n-ary tree
Reverse a number using stack
Reversing the first K elements of a Queue
Reversing a Queue

Check if stack elements are pairwise consecutive
Spaghetti Stack
Interleave the first half of the queue with second half
Remove brackets from an algebraic string containing + and - operators
Growable array based stack
Range Queries for Longest Correct Bracket Subsequence

. Longest Increasing Subsequence Problem