Stack Interview questions

1. Implement Stack basic methods.
2. **Stack from Queues**: How many Queues are required for making a Stack? (Implement it) (Book : 3.4) (Book2: 4.8.12) (Johnson Controls)
3. **Balanced** **Parentheses**: Solve bracket problem ({{{}}]{).(Johnson Controls)
4. **Three in One**: Describe how you could use a single array to implement three stacks. (Book : 3.1) (Book2: 4.8.15, 4.8.16, 4.8.17)
5. **Stack Min**: How would you design a stack which, in addition to push and pop, has a function min which returns the minimum element? Push, pop and min should all operate in 0(1) time. (Book : 3.2)(book2:4.8.6 also 4.8.7)
6. **Stack of Plates**: Imagine a (literal) stack of plates. If the stack gets too high, it might topple.

Therefore, in real life, we would likely start a new stack when the previous stack exceeds some threshold. Implement a data structure SetOfStacks that mimics this. SetOfStacks should be composed of several stacks and should create a new stack once the previous one exceeds capacity.

SetOfStacks. push () and SetOfStacks. pop () should behave identically to a single stack (that is, pop ( ) should return the same values as it would if there were just a single stack).

FOLLOW UP

Implement a function popAt (int index) which performs a pop operation on a specific sub-stack. (Book : 3.3)

1. **Sort Stack**: Write a program to sort a stack such that the smallest items are on the top. You can use an additional temporary stack, but you may not copy the elements into any other data structure (such as an array). The stack supports the following operations: push, pop, peek, and isEmpty.(book : 3.5)(Book2: 4.8.26)
2. **Animal Shelter**: An animal shelter, which holds only dogs and cats, operates on a strictly"first in, first out" basis. People must adopt either the "oldest" (based on arrival time) of all animals at the shelter, or they can select whether they would prefer a dog or a cat (and will receive the oldest animal of that type). They cannot select which specific animal they would like. Create the data structures to maintain this system and implement operations such as enqueue, dequeueAny, dequeueDog, and dequeueCat. You may use the built-in Linked List data structure. (Book : 3.6)
3. **Towers of Hanoi**: In the classic problem of the Towers of Hanoi, you have 3 towers and N disks of different sizes which can slide onto any tower. The puzzle starts with disks sorted in ascending order of size from top to bottom (i.e., each disk sits on top of an even larger one). You have the following

constraints:

(1) Only one disk can be moved at a time.

(2) A disk is slid off the top of one tower onto another tower.

(3) A disk cannot be placed on top of a smaller disk.

Write a program to move the disks from the first tower to the last using stacks.(Book : 8.6)

1. **Stack of Boxes**: You have a stack of n boxes, with widths Wi ' heights hi ' and depths di . The boxes cannot be rotated and can only be stacked on top of one another if each box in the stack is strictly larger than the box above it in width, height, and depth. Implement a method to compute the height of the tallest possible stack. The height of a stack is the sum of the heights of each box. (Book : 8.13)
2. Discuss infix to postfix conversion algorithm using stack.(Book2:4.8.2)
3. For a given array with n symbols how many stack permutations are possible? (Book2:4.8.3)
4. Discuss postfix evaluation using stacks? (Book2:4.8.4)
5. Can we evaluate the infix expression with stacks in one pass? (Book2:4.8.5)
6. 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. (Book2:4.8.8 , 4.8.9 and 4.8.10)
7. Given a stack, how to reverse the contents of the stack using only stack operations (push and pop)? (Book2:4.8.11)
8. 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? (Book2:4.8.14)
9. Consider an empty stack of integers. Let the numbers 1, 2, 3, 4, 5, 6 be pushed on to this stack in the order they appear from left to right. Let 5 indicate a push and X indicate a pop operation. Can they be permuted in to the order 325641(output) and order 154623? (If a permutation is possible give the order string of operations. (Book2:4.8.18)
10. 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? (Book2:4.8.19)
11. Earlier in this chapter, we discussed that for dynamic array implementation of stacks, the ‘repeated doubling’ approach is used. For the same problem, what is the complexity if we create a new array whose size is n + K instead of doubling? (Book2:4.8.20)
12. Given a string containing n S’s and n X’s where 5 indicates a push operation and Vindicates a pop operation, and with the stack initially empty, formulate a rule to check whether a given string 5 of operations is admissible or not? (Book2:4.8.21)
13. **Finding of 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[j + 1]? Another way of asking: Given an array A of integers, find the maximum of j – i subjected to the constraint of A[i] < A[j]. (Book2:4.8.22, 4.8.23)
14. Largest rectangle under histogram: A histogram is a polygon composed of a sequence of rectangles aligned at a common base line. For simplicity, assume that the rectangles have equal widths but may have different heights. For example, the figure on the left shows a histogram that consists of rectangles with the heights 3, 2, 5, 6, 1, 4, 4, measured in units where 1 is the width of the rectangles. Here our problem is: given an array with heights of rectangles (assuming width is 1), we need to find the largest rectangle possible. For the given example, the largest rectangle is the shared part.(Book2:4.8.24)
15. For Error! Reference source not found., can we improve the time complexity? (Book2:4.8.25)
16. 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. (Book2:4.8.27)
17. Recursively remove all adjacent duplicates: Given an array of numbers, recursively remove adjacent duplicate numbers. The output array should not have any adjacent duplicates.

Input: 1,5,6, 8,8,8,0,1,1,0,6,5

Output: 1

Input: 1,9,6, 8,8,8,0,1,1,0,6,5}

Output: 1, 9, 5(Book2:4.8.28)

1. If the stack gets too high, it might overbalance. There-fore, in real life, we would likely start a new stack when the previous stack exceeds some threshold. Implement a data structure that mimics this and composed of several stacks, and should create a new stack once the previous one exceeds capacity. push() and pop() of this class should behave identically to a regular stack. (Book2:4.8.28 part2)