Problem 1

Write a program to reverse a stack using recursion. You are not allowed to use loop constructs like while, for..etc, and you can only use the following ADT functions on Stack S:

- isEmpty(S)
- push(S)
- pop(S)

Hint: The idea of the solution is to hold all values in ***function call stack*** until the stack becomes empty. When the stack becomes empty, insert all held items one by one at the bottom of the stack.

References

- https://www.techiedelight.com/reverse-stack-using-recursion/
- https://afteracademy.com/blog/reverse-a-stack-using-recursion/

Problem 2

Given a string that contains only the following => '{', '}', '(', ')', '[', ']'. At some places there is 'X' in place of any bracket. Determine whether by replacing all 'X's with appropriate bracket, is it possible to make a valid bracket sequence.

Prerequisite: Balanced Parenthesis Expression

Examples
`{X[X([X)}}`
Yes, `{{[]([])}}` is a solution. Hence true.

`[{X}(X)]`
No solution. Hence false.

Hint: When an closing bracket is encountered treat it like the balanced parenthesis expression problem (pop and check for match - if opening is X then consider X as opening bracket for it and **continue using the recursive approach**). If X is encountered, then consider 2 cases (take a copy of the stack for one subproblem) - treat it as opening and hence push to the copy stack and recurse. If this returns true, then there i a solution and return true. If this returns false, then check the case where X is assumed a closing brace, i.e. pop() from the original stack and recurse. If this succeeds we have a solution, else we don't.

References

- https://www.geeksforgeeks.org/balanced-expression-replacement/

Problem 3

Find the largest rectangular area possible in a given histogram where the largest rectangle can be made of a number of contiguous bars. For simplicity, assume that all bars have the same width and the width is 1 unit.

a. Solve it using brute-force (i.e. trying all possible combinations)
Hint: We need to check what is the maximum area for every choice of starting and ending bar (it will be minimum height bar in that range * number of bars). We maintain the bar with maximum area.

b. Solve it using a stack
Hint: The thought process behind this approach is to find the area of the rectangle possible considering each bar

as the bar with minimum height. Now, how will we do this? How to make each bar of minimum height. We can do this if we know which the first bar on the left side of that bar is having less height and similarly which the first bar on the right side is having less height.

References

- https://www.geeksforgeeks.org/largest-rectangular-area-in-a-histogram-using-stack/
- https://afteracademy.com/blog/largest-rectangle-in-a-histogram/

Problem 4

Design a stack that has an additional method `min()` that returns the minimum value remaining in the stack in constant time.

a. Design it using an extra stack
Hint: Maintain an extra stack within the stack. This is used to track the minimum. When pushing an element, check if the pushed value is less than current minimum (top of extra stack), and if so, push it to the extra stack as well. When popping a value, also check if it is the minimum, and if so, pop from the extra stack as well.

b. Design it without using an extra stack
Hint: Maintain the minimum value of the stack in an extra variable. When a value less than the current minimum is being pushed, push instead 2 * value - min, and update min. When popping, if the number is less than the min, then the popped value is the min, and the new min (in remaining stack) is 2 * min - top.

References

- https://www.techiedelight.com/design-stack-which-returns-minimum-element-constant-time/
- https://www.techiedelight.com/design-a-stack-which-returns-minimum-element-without-using-auxiliary-stack/