Stacks and Recursion

1. Redesign the Stack ADT, along with its operations, so that it supports a function `FindMin`, which is $\Theta(1)$.
2. Redesign the functions `EvaluatePostfix` and `InfixToPostfix` to overcome as much limitations as you can, then write a complete program to read an infix string and evaluate it.
3. Write 6 functions for the possible 6 conversions between Infix, Postfix, and Prefix expressions. Which of these functions is easier if written recursively?
4. Write the function `EvaluatePrefix`.
5. Rewrite the functions `EvaluatePrefix` and `EvaluatePostfix` recursively.
6. In mathematical expressions, or any other expression, there can be many parentheses to group together some operations or subparts of the expression. There are three kinds of parentheses: (, {, and [, along with their matching ones:), }, and] respectively. It is required to write a function that takes a string, ending with the null character `'\0'`, and returns 1 if the string has correct matching parentheses and 0 otherwise. For example, the string $[\cdots(\cdots)\{\cdots\}\cdots]$ is valid. However, the string $[\cdots(\cdots\}\{\cdots]$ is not valid because } is the wrong match of (. Also, the string $[\cdots(\cdots)\{\cdots\}]\cdots)$ is not valid, because the last) does not match any parenthesis. Also, the string $(\cdots[\cdots(\cdots)\{\cdots\}]$ is not valid because the first (is not matched. Notice that, the dots in these expressions mean any sequence of characters.

   Hint: Make an empty stack. Read characters until the end of the string. If the character is an open anything, push it onto the stack. If it is a close anything, then if the stack is empty report and error. Otherwise, pop the stack. If the symbol popped is not the corresponding opening symbol, then report an error. At end of the string, if the stack is not empty report an error.
7. Try to write the function recursively; can we? If you can, then draw the recursion tree and see whether you can convert it to iterative version. Is this iterative version the same as the previous function.
8. Write the function `ClearStack` in the implementation level of a linked stack recursively. Draw the recursion tree, and conclude the stack depth. Is there any need for implementing this function recursively than iteratively?