Homework 56 Progress Questions

1. By tracing the discussed algorithm regarding how to convert an infix expression into a postfix expression, please explain in a step-by-step manner how to convert the example infix expression **(((1+2)-(3-4))/(6-5))** to the postfix expression **1 2 + 3 4 - - 6 5 - /**
    a. The first three characters are left parentheses. These will all be pushed to the stack to keep track of operation precedence. Stack is now "(,(,("
    b. The next char is an operand 1, it will be appended to the postfix expression. Postfix is now "1"
    c. The next char is an operator +, the precedence of + and the top stack item are checked, + as current has higher precedence than (on stack, + is simply pushed to the stack. Stack is now "(,(,(,+"
    d. Next char is an operand 2, it is appended to the postfix. Postfix is now "12"
    e. Next char is a right parenthesis, pops from stack to postfix until the top stack item is a left parenthesis, pops "+", pops the left parenthesis as well and discards it. Postfix is now "12+". Stack is now "(,("
    f. Next char is an operator -, checks precedence of – and top stack item, - as current has higher precedence than (on stack, - is pushed to stack. Stack is now "(,(,-"
    g. Next char is a left parenthesis, it is pushed to the stack. Stack is now "(,(,-,("
    h. Next char is an operand 3, it is appended to the postfix. Postfix is now "12+3"
    i. Next char is an operator -, checks precedence of – and top stack item, - as current has higher precedence than (on stack, - is pushed to stack. Stack is now "(,(,-,(,-"
    j. Next char is an operand 4, it is appended to the postfix. Postfix is now "12+34"

k. Next char is a right parenthesis, pops from stack to postfix until the top stack item is a left parenthesis, pops “-“, pops the left parenthesis as well and discards it. Postfix is now “12+34-“. Stack is now “(,(,-“
l. Next char is a right parenthesis, pops from stack to postfix until the top stack item is a left parenthesis, pops “-“, pops the left parenthesis as well and discards it. Postfix is now “12+34--“. Stack is now “(“
m. Next char is an operator /, checks precedence of / and top stack item, / as current has higher precedence than (on stack, / is pushed to stack. Stack is now “(,/”
n. Next char is a left parenthesis, it is pushed to stack. Stack is now “(,/,(“
o. Next char is operand 6, it is appended to the postfix. Postfix is now “12+34--6”
p. Next char is an operator -, checks precedence of – and top stack item, - as current has higher precedence than (on stack, - is pushed to stack. Stack is now “(,/,(,-“
q. Next char is operand 5, it is appended to the postfix. Postfix is now “12+34--65”
r. Next char is a right parenthesis, pops from stack to postfix until the top stack item is a left parenthesis, pops “-“, pops the left parenthesis as well and discards it. Postfix is now “12+34--65-“. Stack is now “(,/”
s. Next char is a right parenthesis, pops from stack to postfix until the top stack item is a left parenthesis, pops “/”, pops the left parenthesis as well and discards it. Postfix is now “12+34--65-/”. Stack is now “”
t. Stack size is 0, no further operations are needed. The postfix expression is complete

2. please evaluate the following postfix expression and write your answer. Show the state of the evaluation stack just before the first arithmetic operation is applied and then just after each additional operation is performed. All operands listed below are single-digit integers. All arithmetic is integer arithmetic.

   Postfix expression: 1 2 + 3 4 - - 6 5 - /

   Your final result should be 4.

   a. First char is operand 1, it is pushed to stack. Stack is now "1"
   b. Next char is operand 2, it is pushed to stack. Stack is now "1,2"
   c. Next char is operator +, stack is popped twice and assigned to variables right and left respectively, operator is identified as add and used to get the result of 3, result is pushed to stack. Stack is now "3"
   d. Next char is operand 3, it is pushed to stack. Stack is now "3,3"
   e. Next char is operand 4, it is pushed to stack. Stack is now "3,3,4"
   f. Next char is operator -, stack is popped twice and assigned to variables right and left respectively, operator is identified as subtract and used to get the result of -1, result is pushed to stack. Stack is now "3,-1"
   g. Next char is operator -, stack is popped twice and assigned to variables right and left respectively, operator is identified as subtract and used to get the result of 4, result is pushed to stack. Stack is now "4"
   h. Next char is operand 6, it is pushed to stack. Stack is now "4,6"
   i. Next char is operand 5, it is pushed to stack. Stack is now "4,6,5"

j. Next char is operator -, stack is popped twice and assigned to variables right and left respectively, operator is identified as subtract and used to get result of 1, result is pushed to stack. Stack is now “4,1”

k. Next char is operator /, stack is popped twice and assigned to variables right and left respectively, operator is identified as integer divide and used to get the result of 4, result is pushed to stack. Stack is now “4”

l. Iterator is now at the end of postfix. Stack size is 1, stack is popped and returned as the result of 4