Lab 03 – Stack Applications

1. **Postfix Evaluation:** Write a JAVA program to evaluate a postfix expression:

    a. Create an empty stack and scan the postfix expression from left to right

    b. If the element is an operand -> push it into the stack

    c. If the element is an operator -> pop twice and get A operator B -> push the result into the stack again

    d. When expression ends, the value in the stack is the final answer

Example: I/P:= 2 10 + 9 6 - / O/P:= 4

2. **Prefix Evaluation:** Repeat Q1 but for Prefix Evaluation

    a. While the steps are very similar to the postfix evaluation (reverse the prefix string to get the postfix notation), you need to be careful of multiple digits – say 12 should not be read as 21 while shifting the characters.

    b. Remember to use any variable, say `shift` that starts as `1` and then is multiplied by `10` .

3. **Infix to Postfix Conversion:** Convert an infix expression to postfix.

    a. Scan the infix expression from left -> right

    b. If the scanned character is an operand, put it into a `postfix` expression

    c. If the scanned character is an operator:

        i. Check priorities of the operator (^ **>** */ **>** +-)

        ii. If the current operator has a higher precedence, then push it into the stack, else, pop all the elements till the precedence of the `tos` **is less than or equal to** the current operator and then push the current operator into the stack.

iii. `^` is right-associative, the rest are left-associative

d. If the scanned character is a `(`, push it to the stack

e. If the scanned character is a `)`, pop the stack until a `(` is encountered. Discard both parenthesis.

f. Repeat Steps (b) to (e) until the entire infix expression is scanned

g. Pop the remaining elements from the stack and add the operators in the postfix expression

Example: I/P:= "a+b*c+d) O/P:= "ab*+d+"

4. **Infix to Prefix Conversion:** Convert a string expression from infix to prefix:

   a. Reverse the infix expression string

   b. Convert the string to **near** postfix (Refer Q3)

      i. Note: For step c -> ii -> **is less than only**

   c. Reverse the postfix expression string

5. **Prefix expression to Postfix:** Convert a prefix expression to postfix:

   a. Reverse the prefix string

   b. If the character is an operand -> push it onto the stack

   c. If the symbol is an operator -> pop two operands from stack -> create a string by concatenating the two operands and the operator -> push the resultant string back to Stack

   d. Repeat Steps (b) to (c) till end of the revesed prefix string is reached