**3. Stacks**

AASTHA SOOD

16BCE1104

Q. 1) Expression evaluator

Assume you are given an arithmetic expression in which only the following operators are allowed: + for addition, - for subtraction, \* for multiplication, ^ for exponentiation and / for division. Make use of the stack data structure and develop an expression evaluator in Python. For simplicity, assume that your arithmetic expression is given in Reverse Polish Notation (see below for what that means). Your evaluator should not only evaluate the expression but must also show changes in the stack as each individual token is processed. For ease, assume that your arithmetic expression deals only with integers. Note: There are three types of notation commonly used for specifying arithmetic expressions. Reverse Polish Notation (RPN) or postfix notation is a mathematical notation in which every operator follows all of its operands, in contrast to Polish or prefix notation which puts the operator before its operands. RPN does not need any parentheses as long as each operator has a fixed number of operands. Infix notation is the standard notation characterized by the placement of operators between operands – "infixed operators" – such as the plus sign in "3 + 4" For example, the expression "5 + ((1 + 2) × 4) − 3" in infix notation (the standard notation) can be written in RPN as below:

5 1 2 + 4 × + 3 –

**ALGORITM**

1) Read Expression in ‘s’

2) Assign stack=[]

3) For i in s.split(‘ ‘):

4) If i in ['+','-','/','\*']:

5) m=stack.pop()

6) n=stack.pop()

7) if i=='-':result=m-n

8) if i=='+':result=m+n

9) if i=='\*':result=m\*n

10) if i=='/':result=m/n

11) stack.append(result)

12) else:

13) stack.append(int(i))

14) Display stack

15) END

16) Display stack.pop()

**PYTHON CODE:**

s=input("Enter the expression : ")

stack=[]

for i in s.split(' '):

if i in ['+','-','/','\*']:

m=stack.pop()

n=stack.pop()

if i=='-':

result=m-n

if i=='+':

result=m+n

if i=='\*':

result=m\*n

if i=='/':

result=m/n

stack.append(result)

else:

stack.append(int(i))

print(stack)

print(stack.pop())

**SCREENSHOTS:**

