Applications of Stacks

- Reversing a list
- Parentheses checker
- **Conversion of an Infix Expression into a Postfix Expression**
- **Evaluation of a Postfix Expression**
- Balancing Symbols
- Function Calls
- Recursion
- Tower of Hanoi

Algebraic Expression Notations

- **Infix, Postfix and Prefix** notations are three different but equivalent notations of writing algebraic expressions.
- **Infix notation** is one in which the operator is placed between the operands. For example, ***A+B***; here, plus operator is placed between the two operands A and B.
- **Prefix notation** is one in which the operator is placed before the operands. For example, ***+AB***; here, plus operator is placed before the two operands A and B.
- **Postfix notation** is one in which the operator is placed after the operands. For example, ***AB+***; here, plus operator is placed after the two operands A and B.

Algebraic Expression Notations

- Although it is easy to write expressions using infix notation, computers find it difficult to parse as they need a lot of information to evaluate the expression.
- Information is needed about operator precedence, associativity rules, and brackets which overrides these rules.
- So, computers work more efficiently with expressions written using prefix and postfix notations.
- Parenthesis-free Prefix notation known as **Polish Notation or PN**
- Parenthesis-free Postfix notation known as **Reverse Polish Notation or RPN.**

Example

- **Infix** - (A + B) * C
- **Prefix** - *+ABC
- **Postfix** - AB+C*
- A **postfix and prefix operation** does not even follow the rules of operator precedence.
- **Postfix -** The operator which occurs first in the expression is operated first on the operands.
- **Prefix -** The operators are applied to the operands that are present immediately on the right of the operator.

Evaluation of an Arithmetic Expression

- **Conversion of an Infix Expression into a Postfix Expression**

and

- **Evaluation of a Postfix Expression**

Infix Expression into a Postfix Expression

Algorithm to convert an Infix notation into postfix notation

```
Step 1: Add ‘)” to the end of the infix expression
Step 2: Push “(“ on to the stack
Step 3: Repeat until each character in the infix notation is scanned
# IF a “(“ is encountered, push it on the stack
# IF an operand is encountered, add it to the postfix expression.
# IF a “)” is encountered, then;
   a)Repeatedly pop from stack and add it to the postfix expression
     until a “(” is encountered.
   b)Discard the “(“. That is, remove the “(“ from stack and do not
     add it to the postfix expression
# IF an operator X is encountered, then;
   c)Repeatedly pop from stack and add each operator (popped from the
     stack) to the postfix expression which has the same precedence or
     a higher precedence than X
    •Push the operator X to the stack
Step 4: EXIT
```

A - (B / C + (D % E * F) / G) * H

Infix Character Scanned	Stack	Postfix Expression
	(	
A	(	A
–	(–	A
(	(– (	A
B	(– (	A B
/	(– (/	A B
C	(– (/	A B C
+	(– (+	A B C /
(	(– (+ (	A B C /
D	(– (+ (	A B C / D
%	(– (+ (%	A B C / D
E	(– (+ (%	A B C / D E
*	(– (+ (*	A B C / D E%
F	(– (+ (*	A B C / D E%F
)	(– (+	A B C / D E%F *
/	(– (+ /	A B C / D E%F *
G	(– (+ /	A B C / D E%F * G
)	(–	A B C / D E%F * G / +
*	(– *	A B C / D E%F * G / +
H	(– *	A B C / D E%F * G / + H
)		A B C / D E%F * G / + H * –

Evaluation of a Postfix Expression

Algorithm to evaluate a postfix expression

```
Step 1: Add a ")" at the end of the postfix expression
Step 2: Scan every character of the postfix expression and
repeat steps 3 and 4 until ")"is encountered
Step 3: IF an operand is encountered, push it on the stack
        IF an operator X is encountered, then
             a)  Pop the top two elements from the stack as A
                 and B
             b)  Evaluate B X A, where A was the topmost
                 element and B was the element below A.
             c)  Push the result of evaluation on the stack
        [END OF IF]
Step 4: SET RESULT equal to the topmost element of the stack
Step 5: EXIT
```

Evaluation of a Postfix Expression

- Let us now take an example that makes use of this algorithm.
- Consider the infix expression given as "**9 - ((3 * 4) + 8) / 4**".
- The infix expression "**9 - ((3 * 4) + 8) / 4**" can be written as "**9 3 4 * 8 + 4 / -**" using postfix notation.
- Look at table which shows the procedure.

Character scanned	Stack
9	9
3	9, 3
4	9, 3, 4
*	9, 12
8	9, 12, 8
+	9, 20
4	9, 20, 4
/	9, 5
–	4