As discussed in [Infix To Postfix Conversion Using Stack](https://www.includehelp.com/c/infix-to-postfix-conversion-using-stack-with-c-program.aspx), the compiler finds it convenient to evaluate an expression in its postfix form. The virtues of postfix form include elimination of parentheses which signify priority of evaluation and the elimination of the need to observe rules of hierarchy, precedence and associativity during evaluation of the expression.

As **Postfix expression** is without parenthesis and can be evaluated as two operands and an operator at a time, this becomes easier for the compiler and the computer to handle.

**Evaluation rule of a Postfix Expression states:**

1. While reading the expression from left to right, push the element in the stack if it is an operand.
2. Pop the two operands from the stack, if the element is an operator and then evaluate it.
3. Push back the result of the evaluation. Repeat it till the end of the expression.

Algorithm

1. **Start**

**2)** Read postfix expression Left to Right  
**3)** If operand is encountered, push it onto Stack [End If]  
**4)** If operator is encountered, Pop two elements  
i) A -> Top element  
ii) B-> Next to Top element  
iii) Evaluate B operator A  
push B operator A onto Stack  
**5)** Set result = pop

**6)End**

**Let's see an example to better understand the algorithm:**

**Expression: 456\*+**

  
