**Algorithm to parse infix expressions and calculate the result**

// Read from input file

// Scan next line to get infix expression and then parse each line

**Covert from infix to postfix method**

**while** InputFile has next line

get next token and trim //to eliminate white space between tokens

**if** token is operand append to postfix string

**else if** the token is open parentheses

**push** token to stack

**else if** token is an operator

**while** stack is not empty

**AND** top of stack is not open parentheses

**AND** precedence of current token is <= precedence of token on top of stack

**Pop** top off the stack and append to the postfix string

**End while**

**Push** current token onto the stack

**Else** (token is closing parentheses)

**While**  top of stack is not open parentheses

**Pop** top of stack and append to the postfix string

**End while**

**Pop** the opening parentheses from the top of stack

**End while**

**Pop** remaining operators off the stack and append to postfix string

**PRECENDENCE METHOD**

**Precedence(string operator)**

**If** operator is “^” return 7

**If** operator is “\*, /, %” return 6

**If** operator is “+” “-“ return 5

**If** operator is “>, >=, <, <=” return 4

**If** operator is “==, !=” return 3

**If** operator is “&&” return 2

**If** operator is “||” return 1

**Throw** illegal argument exception “operator (token) is not supported”

**Evaluate postfix expression**

**While** postfix string has next token

Get the token

**If** the token is an operand

**push** the number onto the stack

**else //token is an operator**

**pop** a number from the stack as the right operand

**pop** a number from the stack as the left operand

evaluate the operation

**if** operator is “/” AND right operand equals zero

**throw** illegal argument exception “Can’t divide by zero”

**if** operator is a Boolean

**if true return 1**

**if false return 0**

**push** result to the stack

**end if**

**end while**

**pop** the stack and return result