**Algorithm convertInfixToPostFix (infix expression)**

// Converts an infix expression to an equivalent postfix expression.

// operatorStack = a new empty stack

// postfix = a new empty string

while (infix has characters left to parse)

{

nextCharacter = next nonblank character of infix //skip spaces

switch (nextCharacter)//

{

case operand: Append nextCharacter to postfix

break

case '^' : operatorStack.push(nextCharacter)

break

case '+' :

case '-' :

case '\*' :

case '/' : while (!operatorStack.isEmpty() and

precedence(nextCharacter) <=

precedence(operatorStack.getTop ())

{

Append operatorStack.getTop() to postfix

operatorStack.pop()

}

operatorStack.push(nextCharacter)

break

case '(' : operatorStack.push(nextCharacter)

break

case ')' : // stack is not empty if infix expression is valid

topOperator = operatorStack.getTop()

while (topOperator != '(')

{

Append topOperator to postfix

operatorStack.pop()

topOperator = operatorStack.getTop()

}

operatorStack.pop()

break

default : break

}

}

//append to postfix expression the operators remaining in the stack

while (!operatorStack.isEmpty())

{

topOperator = operatorStack.getTop()

Append topOperator to postfix

operatorStack.pop()

}

return postfix