**Algorithm evaluateInfix (infix expression)**

**Algorithm evaluateInfix (infix)**

**// Evaluates an infix expression.**

**operatorStack = a new empty stack**

**valueStack = a new empty stack**

**while (infix has characters left to process)**

**{**

**nextCharacter = next nonblank character of infix**

**switch (nextCharacter)**

**{**

**case operand:**

**valueStack.push (value of the variable nextCharacter)**

**break**

**case '^':**

**operatorStack.push (nextCharacter)**

**break**

**case '+':**

**case '-':**

**case '\*':**

**case '/':**

**while (!operatorStack.isEmpty () and**

**precedence(nextCharacter) <= precedence**

**(operatorStack.getTop ()))**

**{**

**// Execute operator at top of operatorStack**

**topOperator = operatorStack.getTop ()**

**operatorStack.pop()**

**operandTwo = valueStack.getTop ()**

**valueStack.pop()**

**operandOne = valueStack.getTop()**

**valueStack.pop()**

**result = the result of the operation in**

**topOperator and its operands**

**operandOne and operandTwo**

**valueStack.push (result)**

**}**

**operatorStack.push (nextCharacter)**

**break**

**case '(':**

**operatorStack.push (nextCharacter)**

**break**

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

**//valid**

**topOperator = operatorStack.getTop ()**

**operatorStack.pop()**

**while (topOperator != '(')**

**{**

**operandTwo = valueStack.getTop ()**

**valueStack.pop()**

**operandOne = valueStack.getTop ()**

**valueStack.pop()**

**result = the result of the operation in**

**topOperator and its operands**

**operandOne and operandTwo**

**valueStack.push (result)**

**topOperator = operatorStack.getTop ()**

**operatorStack.pop()**

**}**

**break**

**default:**

**break**

**}**

**}**

**while (!operatorStack.isEmpty ())**

**{**

**topOperator = operatorStack.getTop()**

**operatorStack.pop()**

**operandTwo = valueStack.getTop ()**

**valueStack.pop()**

**operandOne = valueStack.getTop ()**

**valueStack.pop()**

**result = the result of the operation in topOperator and its**

**operand operandOne and operandTwo**

**valueStack.push (result)**

**}**

**return valueStack.getTop ()**