**SSN College of Engineering, Kalavakkam**

**Department of Computer Science and Engineering**

**III Semester - CSE**

# UCS 1312 Data Structures Lab Laboratory

|  |  |
| --- | --- |
| **Academic Year: 2019-2020** | **Batch: 2018-2022** |

**Exercise 6: Construction and Evaluation of Expression Tree**

* Get an expression and convert that into postfix notation using the conversion algorithm
* Using stack, construct expression tree from the postfix notation

while(not the end of the expression)

{

if(the next symbol in the expression is an operand)

{

create a node for the operand ;

push the reference to the created node onto the stack ;

}

if(the next symbol in the expression is a binary operator)

{

create a node for the operator ;

pop from the stack a reference to an operand ;

make the operand the right subtree of the operator node ;

pop from the stack a reference to an operand ;

make the operand the left subtree of the operator node ;

push the reference to the operator node onto the stack ;

}

}

* Evaluate the expression from evaluation tree using the algorithm

Evaluate(ExpressionTree t)

{

if(t is a leaf)

return value of t's operand ;

else

{

operator = t.element ;

operand1 = evaluate(t.left) ;

operand2 = evaluate(t.right) ;

return(applyOperator(operand1, operator, operand2) ;

}

}

Note:

1. Create Expression tree ADT with the members expression, postfix and value. It has the following functions
   1. void InfixtoPostfix(ExpTree \*t)
   2. void diplay(ExpTree \*t) – displays the infix expression, postfix and the value (initially it will be 0)
   3. void evaluation(Expression \*t)