CSE 3010 – Data Structures & Algorithms

Lecture #11

What will be covered today

- Evaluation of arithmetic expression – Example to illustrate Stack data structure
- Representation of a linked node
- Creating a linked node

Evaluation of expressions by a compiler

- Step 1: Convert from infix form to postfix form

```
a + b = ab+
a + b * c = abc*+
a + b / c * d = abc/d*-
(a + b) * c = ab+c*
```

- Step 2: Evaluate the expression in the postfix form

```
ab+
ab*+
abc/d*-
ab+c*
```

Step 1: Convert infix expression to postfix expression

Input: Infix expression as an array of characters

Output: Postfix expression as an array of characters

1. Check characters of the input from left to right one by one
    a. If the character is an operand - add it to the output
    b. If the character is left parenthesis - push it on the stack
    c. If the character is an operator
        a. Pop the operator that has same or higher precedence – add it to the output
        b. If the operator has higher preference, push the operator into the stack
    d. If the character is right parenthesis
        a. Pop all the operators up to the left parenthesis
        b. Add popped operators to the output
        c. Pop the left parenthesis
2. Pop remaining elements from the stack when the last character of the input is checked

Evaluate the postfix expression

Input: Postfix expression as an array of characters

Output: Value of the expression

1. Check characters of the input from left to right one by one
   a. If the character is an operand – push into the stack
   b. If the character is an operator
      a. Pop the required operands (based on unary, binary or ternary) from the stack
      b. Evaluate using the operator and the popped operands from the stack
      c. Push the result into the stack
2. Continue steps in (1) above until all characters of the input are scanned
3. Value inside the stack at the end of the input is the value of the expression