Stack as Subclass of Linked List

LINKED_LIST

- Member variable for headPtr
- Member functions:
    - **constructor**
    - **destructor**
    - **insertAtHead, insert**
    - **removeAtHead, remove**
    - **clearContents**
    - **isEmpty**

STACK

- Can inherit from **LINKED_LIST**
- No additional member variables
- Member functions:
    - **constructor**
    - **destructor**
    - **push**
    - **pop**
    - **top**

Arithmetic Expression Formats

Infix

- Operator **between** operands
- Ex: 3 + 4

Prefix

- Operator **before** operands
- Ex: + 3 4

Postfix

- Operator **after** operands
- Ex: 3 4 +

Using Stacks to Evaluate Arithmetic Expressions

Input: fully parenthesized correctly formed **infix** expression, each number non-negative

Output: value of the expression

Design:

- Use 2 stacks: one for NUMBERS from input expression and subresults, another stack for OPERATIONS that need to be performed.
- When number encountered in input, push onto NUMBERS stack.
- When operation character (+, -, *, /) encountered, push onto OPERATIONS stack.
- When right parenthesis, pop 2 numbers from NUMBERS stack and pop 1 operator from OPERATIONS stack. Perform operation and push result on NUMBERS stack.
- When left parenthesis or blank, throw away (we're assuming parens balanced)
- Halt when end of expression, at which point single number (answer) on NUMBERS stack.

Example: (((5 * 4) / 2) * (61 – 58))