Reverse Polish Notation (RPN) Homework Name: Taylan Unal

The algorithm for evaluating any postfix expression is fairly straightforward:

* While there are input tokens left
  + Read the next token from input.
  + If the token is a value
    - Push it onto the stack.
  + Otherwise, the token is an operator (operator here includes both operators, and functions).
    - It is known [*a priori*](http://en.wikipedia.org/wiki/A_priori) that the operator takes **n** arguments.
    - If there are fewer than **n** values on the stack
      * **(Error)** The user has not input sufficient values in the expression.
    - Else, Pop the top **n** values from the stack.
    - Evaluate the operator, with the values as arguments.
    - Push the returned results, if any, back onto the stack.
* If there is only one value in the stack
  + That value is the result of the calculation.
* If there are more values in the stack
  + **(Error)** The user input has too many values.

Evaluate the following expressions using the algorithm above:

1. 3 4 + [7]
2. 3 4 5 \* − [-17]
3. 6 3 / [2]
4. 5 1 2 + 4 \* + 3 − [14]
5. 1 2 + 4 \* 5 + 3 − [14]
6. 3 5 \* 4 2 - 4 \* + [68]
   1. Write each of the problems above in standard notation with parenthesis to show order of operations:
      1. (3+4)
      2. 3- (4\*5)
      3. (6/3)
      4. (5 + ((1+2) \* 4)) – 3
      5. (((1+2) \* 4) + 5) - 3
      6. ((3\*5) + (4-2)) \*4