**Homework 1 - Calculator grammar**

For the first homework you should implement a simple calculator. The calculator should accept expressions with addition, subtraction, multiplication, and division operators, as well as parentheses. The grammar (for single-digit numbers) is summarized in:   
  
exp  ->   num | exp op exp | (exp)   
op    ->   + | - | \* | /   
num ->   0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9   
  
You need to change this grammar to support priority between addition and multiplication, to remove the left recursion for LL parsing, etc. The homework deliverables are divided into 3 parts.   
  
1. You have to write (and provide in a .txt of .pdf file) the FIRST & FOLLOW sets for the LL(1) version of the above grammar. In the end you will summarize them in a single lookahead table (include a row for **every** derivation in your final grammar).   
  
2. Based on your LL(1) grammar, you have to write a recursive descent parser in Java that reads expressions from **standard input** and prints them in [prefix notation](http://en.wikipedia.org/wiki/Polish_notation), where every sub-expression is surrounded with parentheses\*, or prints "parse error" if there is a syntax error. You don't need to identify blank space or multi-digit numbers. You can read the symbols one by one (as in the C getchar function). The expression must end with a newline or EOF.   
\*That is, the expression "3 \* 5 + 5 - 4 - 3" must be transformed to " (+ (\* 3 5) (- (- 5 4) 3))"  
  
3. You will write a full LR grammar for a calculator using sableCC. The calculator must support the 4 operations, but here you have to do normal parsing, i.e., to ignore blank space and receive non-negative integers of any length. This program will only recognize whether the expression is correct, and will not process it further.