# Reflection: ExpressionEvaluator: Arnav Krishnan

What I learned from this project that I found most interesting was our method of evaluating expressions, from left to right, using operator precedence. I thought this was interesting as the algorithmic approach doesn't really match the way I would evaluate an expression- I would start with the 'nested-most' parens, and work going from inside the parens to out. I thought our paren logic was quite neat and I really liked the table/chart you presented based on what our curToken is (whether we evaluate, add to stack, error, etc.). I have many questions that I still wonder about after this project, with the main 3 being:

1. How might we adapt this expression evaluator to use variables (such as (x+1)(3x+2) =3x^2+5x+2)? I know it would be very difficult, especially if we let the user use any letter as a variable.

2. Is this 2-stack approach the same way programming languages, such as java, evaluate expressions? Do they use recursion in their implementation, and could we?

3. How would our implementation change if we used pre/postfix instead of infix? What if we allowed functions such as sin() or sqrt()?