Assignment 2

Jacob Ewen

ULID: jewen

Secret name: twoewen

**Data types used:**

StringTokenizer tokens

This was used to store the complete list of tokens that were generated from the input string.

Stack<String> tokenStack

This stores the token types of what is in the work stack (“n”, “E”, “T”, “+”,. “$”, etc.)

Stack<Double> numStack

This stack stores the values of the tokens n, E, T, and F in order.

Stack<Integer> stateStack

This stack stores the row values of the corresponding elements of tokenStack.

Boolean valid

This is what I set to true if the expression was valid.

**Methods:**

void Chomp()

This method pushes the next token from the StringTokenizer onto the tokenStack, and checks if it is a number.

Void shift(int state)

This takes pushes the state integer onto the stateStack, and calls chomp().

Void reduce(int rule)

This handles the reduction step with nested switch/case statements. The rule integer corresponds to different reduction rules. It will pop off the correct amount of tokens from tokenStack and numStack, then calculate the new value and push it to numStack and tokenStack. Then it calls reduceState

Void reduceState(int num)

This handles the state changes when you do the reduction step (the part where you check the far right column.)

void iterate()

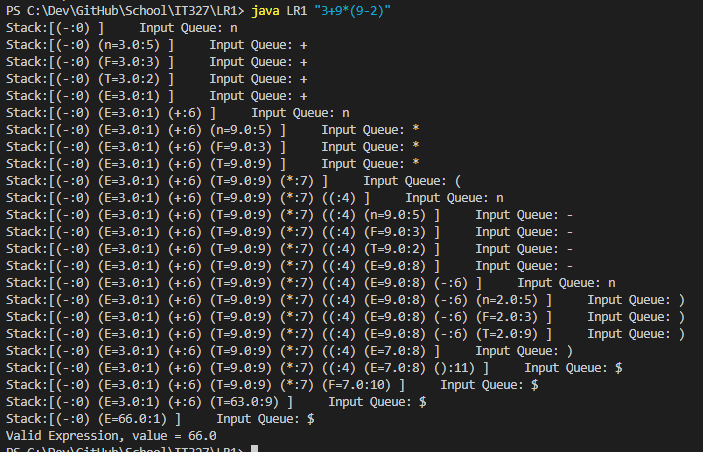
This is essentially a switch/case statement that corresponds to the parsing chart. It checks the next token and the top of the stateStack to do the correct action.

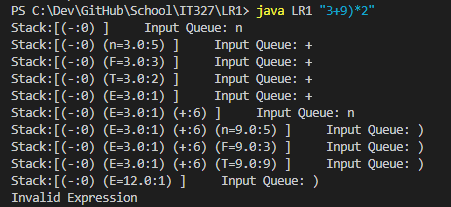
Void main()

This initializes some important variables, gets the input, and calls iterate to start it all.

**Issues and Difficulties:**

Honestly I feel as if this was easier to implement than the LL(1) parser, as most of the logic is contained in the parsing table, which I just had to copy. Any issues I had was simply input errors when I was filling out the switch/case in *Iterate()*. However due to the limitations of the StringTokenizer class I wasn’t able to print all of the input queue for the debugging.



**Outputs:**