HW4 - Programming Languages

Kyle Beard, Giulia Lorini, Troy Tully

Tokens:
Number: [0-9]+(\.[0-9]+)?
String: \“.*\“
Identifier: [a-zA-Z_][0-9a-zA-Z_]*
BinaryOperator: [+\-*/^]
Plus: \+
Divide: \/
Times: *
Minus: -
Power: \^
Sin: sin
Cos: cos
Tan: tan
SQRT: sqrt
Print: print
Input: input
Parentheses: \(\)
SemiColon: ;
Comma: ,
Equals: =

2. (30%) Write down a grammar in BNF or EBNF form for this language. Use the token types from #1 as the terminals in your language.

```
statement ::= print SEMICOLON
        | assignment SEMICOLON
        | print SEMICOLON statement
        | assignment SEMICOLON statement

print ::= PRINT expr
        | PRINT string

string ::= STRING_LITERAL
        | STRING_LITERAL COMMA string
        | expr COMMA string

assignment ::= IDENTIFIER EQUALS expr
        | IDENTIFIER EQUALS input
```

```
input ::= INPUT string
        | INPUT expr

expr ::= expr:a PLUS term
        | expr:a MINUS term
        | term

term ::= term:a TIMES negation
        | term:a DIVIDE negation
        | negation

negation ::= MINUS power
        | power

power ::= func POWER power
        | func

func ::= SIN LPAREN expr RPAREN
        | COS LPAREN expr RPAREN
        | TAN LPAREN expr RPAREN
        | SQRT LPAREN expr RPAREN
        | paren

paren ::= LPAREN expr:a RPAREN
        | NUMBER
        | IDENTIFIER
```