BNF Rules

```
Program ::= Funcs
 Funcs ::= Func
 Funcs ::= Func Functions
  Func ::= Type Ident "(" [Params] ")" "{"
     Stms "}"
   Stms ::= Stm [illegible] Stms
     Stms ::= NULL
   Type ::= IntType | FloatType | StringType
     Params ::= Param
     Params ::= Param "," Params
       Param ::= Type Ident

Stm
 SDec. Stm ::= Type Ident [ "=" Exp ] ";"
SAssign. Stm ::= Ident "=" Exp ";"
SIf. Stm ::= "if" "(" Exp ")" BlockStms IRest
  REmp. IRest ::= NULL
  RElse. IRest ::= "else" BlockStms
 RElseIf. IRest ::= "else" "if" "(" Exp ")"
                    BlockStms IRest
 BlockStms ::= "{" Stms "}"
  BlocStms ::= Stm
```

SWhile: Stm ::= "while" "(" Exp ")" BlockStm

SFor. Stm ::= "for" "(" [~~Type~~ Ident] ";" [Exp] ";"
SAssign
Ident["=" Exp]
SAssign
[Ident "=" Exp] ")"
Block Stms

SReturn. Stm ::= "Return" Exp ";"

SExp. Stm ::= Exp ";"

Exp ::= Exp1 R Exp

RExp ::= BinComp Exp1 RExp

RExp ::= NULL

Exp1 ::= Exp2 RExp1

RExp1 ::= "+" Exp2 RExp1

RExp1 ::= "-" Exp2 RExp1

RExp1 ::= NULL

Exp2 ::= Exp3 RExp2

RExp2 ::= "*" Exp3 RExp2

RExp2 ::= "/" Exp3 RExp2

RExp2 ::= NULL

Exp 3 ::= NumTypes | Ident

BExp3. Exp3 ::= "(" Exp ")"

FExp3. Exp3 ::= FuncCall

BinComp ::= ">" | "<" | ">=" | "<=" | "=="

Each of these is token. For example GreaterEqual token
that is why using ~~Rest~~ is unnecessary

```
NumTypes ::= Int | ~~Double~~ Float
FuncCall ::= Ident '(' [Args] ')'
  Args ::= Arg
  Args ::= Arg "," Args
    Arg ::= Exp
      ~~Arg ::= Ident~~

Stm.
 SBreak.    Stm ::= "break" ";"
 SContinue. Stm ::= "continue" ";"
```