BNF Rules

```
Pdefs. Program ::= [FuncDef];
terminator FuncDef "";
comment "//";
comment "/*" "*/";

DFun. FuncDef ::= Type Id "(" [Param]
")" "{" [Stm] "}";
Separator Param ",";
terminator Stm "";

ADecl. Param ::= Type Id;

SExp. Stm ::= Exp ";";
SDecl. Stm ::= Type Id ";";
SDecls. Stm ::= Type Id "," [Id] ";";
SAssign. Stm ::= Id "=" Id ";" | Id
"=" Exp ";";
SInit. Stm ::= Type Id "=" Exp ";";
SReturn. Stm ::= "return" Exp ";";
SWhile. Stm ::= "while" "(" Exp ")"
Stm;
SBlock. Stm ::= "{" [Stm] "}";
SIf. Stm ::= "if" "(" Exp ")" Stm
IRest;
IRest. IRest ::= "else" Stm | null;

Exp ::= Exp1RExp;
RExp ::= BinCompExp1RExp;
RExp ::= NULL;
```

```
Exp1 ::= Exp2RExp1;
RExp1 ::= "+" Exp2RExp1;
RExp1 ::= "-" Exp2RExp1;
RExp1 ::= NULL;
Exp2 ::= Exp3RExp2;
RExp2 ::= "*" Exp3RExp2;
RExp2 ::= "/" Exp3RExp2;
RExp2 ::= NULL;
Exp3 ::= NumTypes | Id;
Exp3 ::= "(" Exp ")";
Exp3 ::= FuncCall;
BinComp ::= ">" | "<" | ">=" | "<=" |
"==";
FuncCall ::= Id "(" [Exp] ")";
NumTypes ::= Integer | Float;

coercions Exp 3;
separator Exp ",";

INTType. Type ::= "int";
FLOATType. Type ::= "float";
STRINGType. Type ::= "string";

token Id (letter (letter | digit |
'_')*);
```