Compiler Construction Stage 1

Group 1		Group 2	
Divyanshu Goyal	2011B4A7795P	Siddharth Bhatia	2011B4A7680P
Abhishek Modi	2011B4A7331P	Akshay Aurora	2011B4A7658P

Language Description

The language is a **general purpose language** with neat and powerful constructs. It supports **structured programming**, **lexical variable scope** and **recursion** along with **dynamic type inference**. In addition the language provides **special support for String and Grid** data structures to handle character arrays and 2-dimensional integer arrays easily. The language also features a **mutable construct** which enables the programmer to declare a variable mutable/immutable i.e. allow/disallow change after first initialization (By default all variables are immutable).

Language Features

- Data Types
  - `i32, f32, char, boolean, string`
- Type inference
  - Data types can be initialised without explicitly mentioning data type
- Mutability
  - By default, all variables are immutable. To initialise variables as mutable - `mut` modifier can be used.
- Multiple assignments
  - Multiple variables can be initialised in single statement
- Strings
- Functions
  - Support for Overloading
- Conditional statements if-else
- Iterative statements
  - Support for while statements with break
- Arithmetic operators (+ - * /)
  - Defined on i32 and f32
- Logical operators (and or not)
  - Defined on boolean expressions
- Relational operators (> < = <= >= <> ==)
  - Defined for i32, f32 and strings

Tokens

Pattern	Token	Purpose
=	ASSIGNOP	Assignment operator
//[^\n]	COMMENT	Comment
[a-z|A-Z] [a-zA-Z0-9_]*	ID	Identifier (used as Variables)
[0-9][0-9]*	NUM	Integer number
[0-9][0-9]*.[0-9][0-9]	FLOAT	Real number
["][a-z][a-z]*["]	STRL	String literal
['][a-z][']	CHARL	Char literal
return	RETURN	**Keyword** return
char	CHAR	**keyword** char
i32	I32	**Keyword** int
f32	F32	**Keyword** real
bool	BOOL	**Keyword** bool
string	STRING	**Keyword** string
main	MAIN	**Keyword** main
fn	FN	**Keyword** fn
let	LET	**Keyword** let
while	WHILE	**Keyword** while
break	BREAK	**Keyword** break
[	OSQUARE	Left Square bracket
]	CSQUARE	Right Square bracket
(	OPAREN	Open parenthesis
)	CPAREN	Close parenthesis
{	OBRACE	Open braces
}	CBRACE	Close braces

;	SEMICOLON	Semicolon as separator
:	COLON	Colon
,	COMMA	Comma
if	IF	**Keyword** if
else	ELSE	**Keyword** else
elseif	ELSEIF	**Keyword** elseif
scan	SCAN	**Keyword** scan
print	PRINT	**Keyword** print
+	PLUS	Addition operator
-	MINUS	Subtraction operator
*	MUL	Multiplication operator
/	DIV	Division operator
and	AND	Logical and
or	OR	Logical or
not	NOT	Logical not
<	LT	Relational operator less than
<=	LE	Relational operator less than or equal to
==	EQ	Relational operator equal to
>	GT	Relational operator greater than
>=	GE	Relational operator greater than or equal to
<>	NE	Relational operator not equal to
->	RARROW	Return Type beginning
.	DOT	Method beginning

Grammer

```
Grammar = (NonTerminals, Terminals, Rules, Start)

Start = Program

NonTerminals = {Program, Functions, FunctionDef, fnReturn, Statements,
moreStmts, Stmt, ReturnStmt, BreakStmt, DeclarationStmt, moreDeclarations,
mutMod, Declaration, moreTypes, AssignStmtType2, listTypes, typeList,
moreList, singleAssn, multAssn, moreAssn, IDStmts, IDStmts2, Index,
moreIndex, AssignStmtType1, FunCall, MethodCall, FunCallStmt, MethodStmt,
Type, parameterList, remainingList, IfStmt, ElseStmt, IStmt, OStmt, value,
array, IDList, moreIds, arithExpn, moreTerms, arithTerm, moreFactors,
factor, opLow, relType, opHigh, boolExpn, logicalOp, relationalOp, LoopStmt,
grid, rows, moreRows, row, moreNums, boolean}

Terminals = {MAIN, OPAREN, CPAREN, OBRACE, CBRACE, FN, ID, COMMENT, RETURN,
SEMICOLON, Break, LET, MUT, ASSIGNOP, COMMA, COLON, OSQUARE, CSQUARE, NUM,
DOT, BOOL, F32, I32, string, type, IF, ELSEIF, ELSE, SCAN, PRINT, CHARL,
STRINGL, MINUS, PLUS, DIV, MUL, AND, NOT, OR, EQ, GT, GTE, LT, LTE, NE,
WHILE, FALSE, TRUE}

Rules = {
   Program          → Functions MAIN OPAREN CPAREN OBRACE Statements CBRACE
   Functions        → FunctionDef Functions | ε
   FunctionDef      → FN ID OPAREN parameterList CPAREN fnReturn OBRACE
                      Statements CBRACE
   fnReturn         → -> Type | ε
   Statements       → Stmt moreStmts
   moreStmts        → Stmt moreStmts | ε
   Stmt             → COMMENT | ID IDStmts | IfStmt | IStmt | BreakStmt |
                      ReturnStmt | LoopStmt | OStmt | DeclarationStmt
                      | AssignStmtType2
   ReturnStmt       → RETURN relType SEMICOLON
   BreakStmt        → Break SEMICOLON
   DeclarationStmt  → LET mutMod moreDeclarations SEMICOLON
   moreDeclarations → Declaration | AssignStmtType2
   mutMod           → MUT | ε
   Declaration      → ID ASSIGNOP arithExpn moreTypes
   moreTypes        → COMMA Declaration | ε
```

```
AssignStmtType2  → listTypes ASSIGNOP OPAREN multAssn CPAREN SEMICOLON
listTypes        → OPAREN typeList CPAREN
typeList         → ID moreList
moreList         → COMMA typeList | ε
singleAssn       → arithExpn
multAssn         → singleAssn moreAssn
moreAssn         → COLON multAssn | COMMA multAssn | ε
IDStmts          → OSQUARE ID CSQUARE OSQUARE ID CSQUARE AssignStmtType1
                  | FunCallStmt | AssignStmtType1 | MethodStmt
IDStmts2         → FunCall | MethodCall | Index | ε
Index            → OSQUARE ID CSQUARE moreIndex | ID | NUM
moreIndex        → OSQUARE ID CSQUARE | ε
AssignStmtType1  → ASSIGNOP singleAssn SEMICOLON
FunCall          → OPAREN IDList CPAREN
MethodCall       → DOT ID FunCall
FunCallStmt      → FunCall SEMICOLON
MethodStmt       → MethodCall SEMICOLON
Type             → BOOL | F32 | I32 | string
parameterList    → ID COLON type remainingList
remainingList    → COMMA parameterList | ε
IfStmt           → IF OPAREN boolExpn CPAREN OBRACE Statements CBRACE
                   ElseStmt
ElseStmt         → ELSEIF OPAREN boolExpn CPAREN OBRACE Statements
                   CBRACE ElseStmt | ELSE OBRACE Statements CBRACE | ε
IStmt            → SCAN OPAREN ID CPAREN SEMICOLON
OStmt            → PRINT OPAREN ID CPAREN SEMICOLON
value            → CHARL | NUM | STRINGL | boolean | array | grid
array            → OSQUARE multAssn CSQUARE
IDList           → ID moreIds | ε
moreIds          → COMMA IDList | ε
arithExpn        → arithTerm moreTerms
moreTerms        → opLow arithExpn | ε
arithTerm        → factor moreFactors
moreFactors      → opHigh arithTerm | ε
factor           → OPAREN arithExpn CPAREN | relType
opLow            → MINUS | PLUS
relType          → ID IDStmts2 | value
opHigh           → DIV | MUL
boolExpn         → OPAREN boolExpn CPAREN logicalOp OPAREN boolExpn
                   CPAREN | relType relationalOp relType
```

```
    logicalOp        → AND | NOT | OR
    relationalOp     → EQ | GT | GTE | LT | LTE | NE
    LoopStmt         → WHILE OPAREN boolExpn CPAREN OBRACE Statements CBRACE
    grid             → OBRACE rows CBRACE
    rows             → row moreRows
    moreRows         → COLON rows | ε
    row              → NUM moreNums
    moreNums         → COMMA NUM moreNums | ε
    boolean          → FALSE | TRUE
}
```

Test Cases

```
// #1  Sum upto n numbers
// Features - loop, I/O, multiple-assignment

fn main() {
     let mut (i, n, sum) = (1, 0, 0);
     scan(n);
     while(i < n) {
         sum = sum +  i;
          i = i + 1;
     }
     print(sum);
}
```

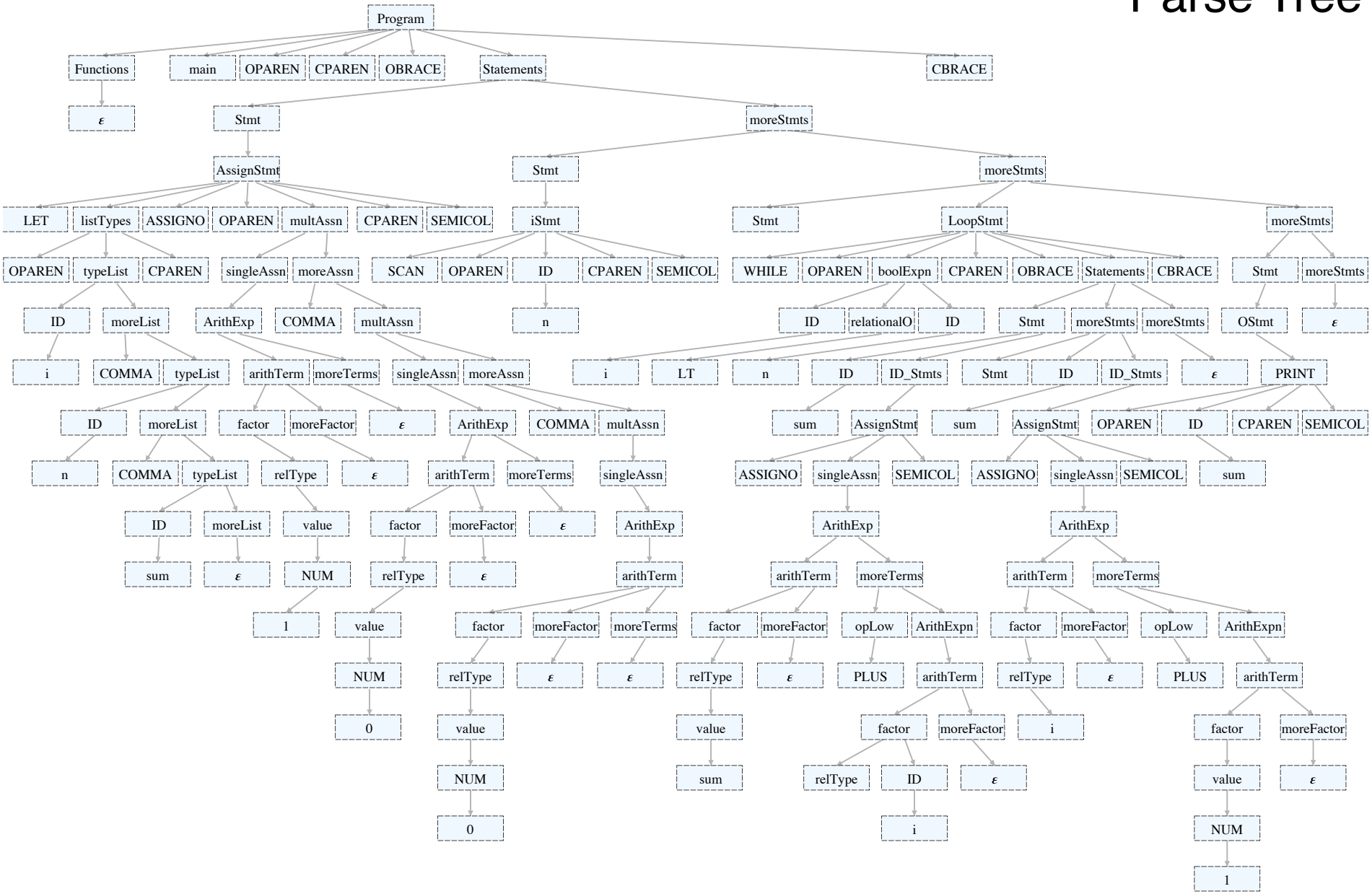
```
// #2 Check if string is Palindrome
// Features - loop, string, boolean

fn main() {
     let mut str1 = "radar";
     if(str1 == str1.reverse()){
         print("Yes");
      }
}
```

```
// #3 Add using functions
// Features - loop, I/O, if, functions

fn add(x:i32, y:i32) -> i32 {
    let mut ans = 0;
    ans = x + y;
    return ans;
}

fn main (){
    let mut x = 0;
    x = add(1, 2);
    print(x);
}
```

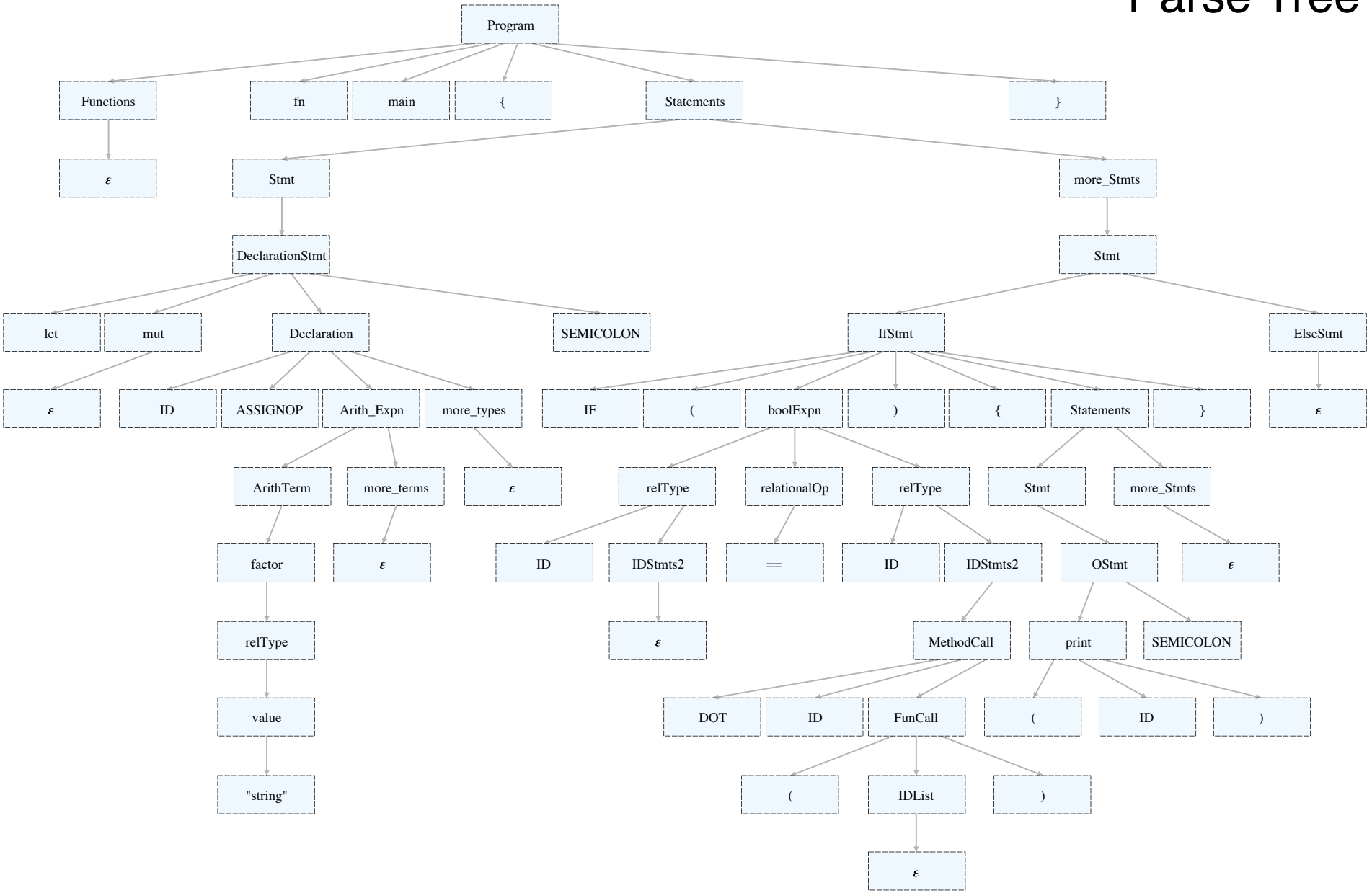
```
// #4 Find minimum element in array
// Features - loop, arrays

fn main() {
     let arr = [1, 1, 2, 5];
      let mut i = 1;
     let mut min = arr[0];
     while(i < arr.size()) {
          if(arr[i] < min) {
            min = arr[i];
          }
          i++;
     }

}
```

```
// #5 Print String in uppercase
// Feature String

fn  main(){
    let mut s = "Hello";
    s.toUpper();
    print(s);
}
```

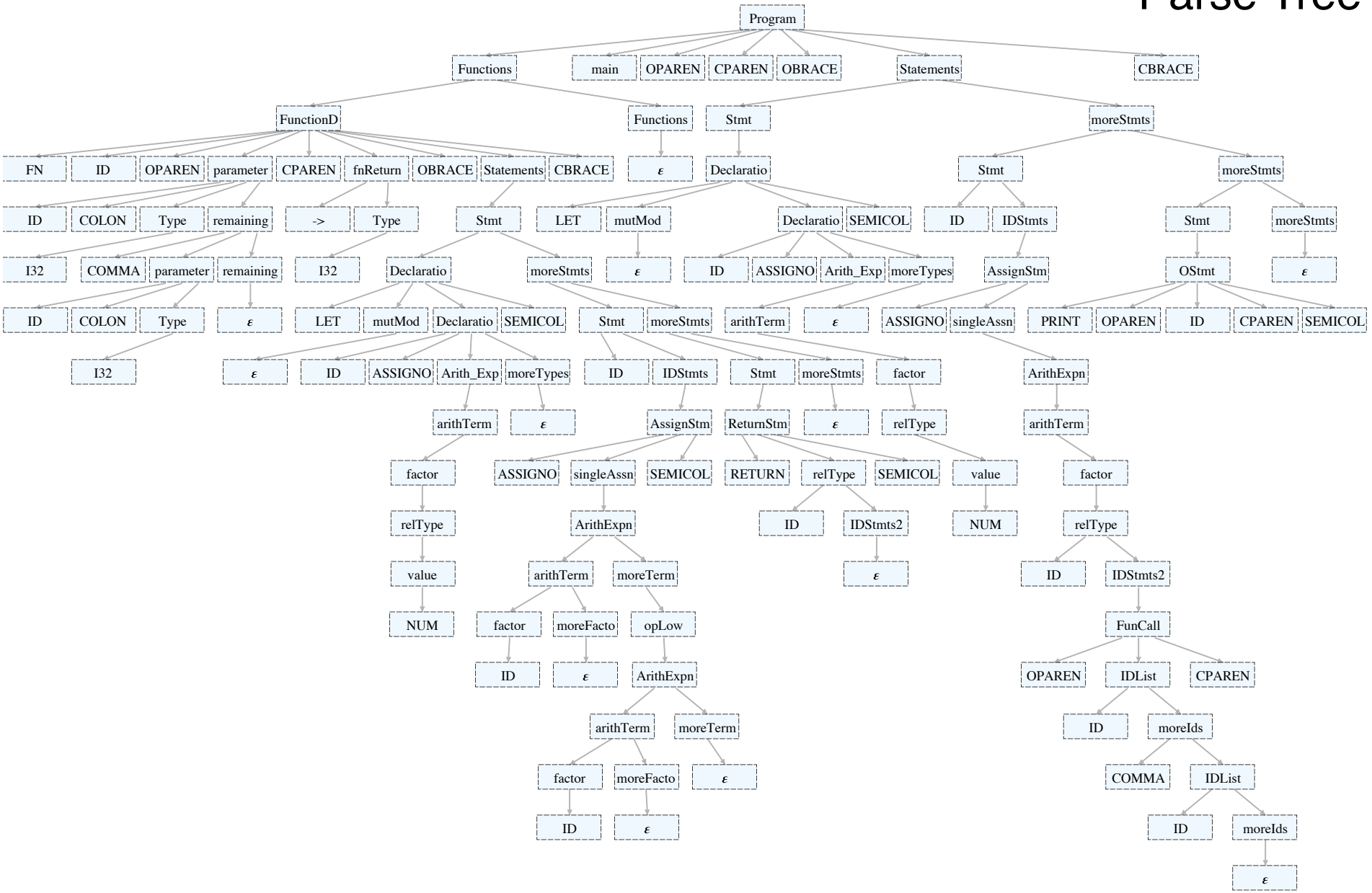
Parse Tree 1


Program
Functions
main
OPAREN
CPAREN
OBRACE
Statements
CBRACE
ε
Stmt
moreStmts
AssignStmt
Stmt
moreStmts
LET
listTypes
ASSIGNO
OPAREN
multAssn
CPAREN
SEMICOL
iStmt
Stmt
LoopStmt
moreStmts
OPAREN
typeList
CPAREN
singleAssn
moreAssn
SCAN
OPAREN
ID
CPAREN
SEMICOL
WHILE
OPAREN
boolExpn
CPAREN
OBRACE
Statements
CBRACE
Stmt
moreStmts
ID
moreList
ArithExp
COMMA
multAssn
n
ID
relationalO
ID
Stmt
moreStmts
moreStmts
OStmt
ε
i
COMMA
typeList
arithTerm
moreTerms
singleAssn
moreAssn
i
LT
n
ID
ID_Stmts
Stmt
ID
ID_Stmts
ε
PRINT
ID
moreList
factor
moreFactor
ε
ArithExp
COMMA
multAssn
sum
AssignStmt
sum
AssignStmt
OPAREN
ID
CPAREN
SEMICOL
n
COMMA
typeList
relType
ε
arithTerm
moreTerms
singleAssn
ASSIGNO
singleAssn
SEMICOL
ASSIGNO
singleAssn
SEMICOL
sum
ID
moreList
value
factor
moreFactor
ε
ArithExp
ArithExp
ArithExp
sum
ε
NUM
relType
ε
arithTerm
arithTerm
moreTerms
arithTerm
moreTerms
1
value
factor
moreFactor
moreTerms
factor
moreFactor
opLow
ArithExpn
factor
moreFactor
opLow
ArithExpn
NUM
relType
ε
ε
relType
ε
PLUS
arithTerm
relType
ε
PLUS
arithTerm
0
value
value
factor
moreFactor
i
factor
moreFactor
NUM
sum
relType
ID
ε
value
ε
0
i
NUM
1

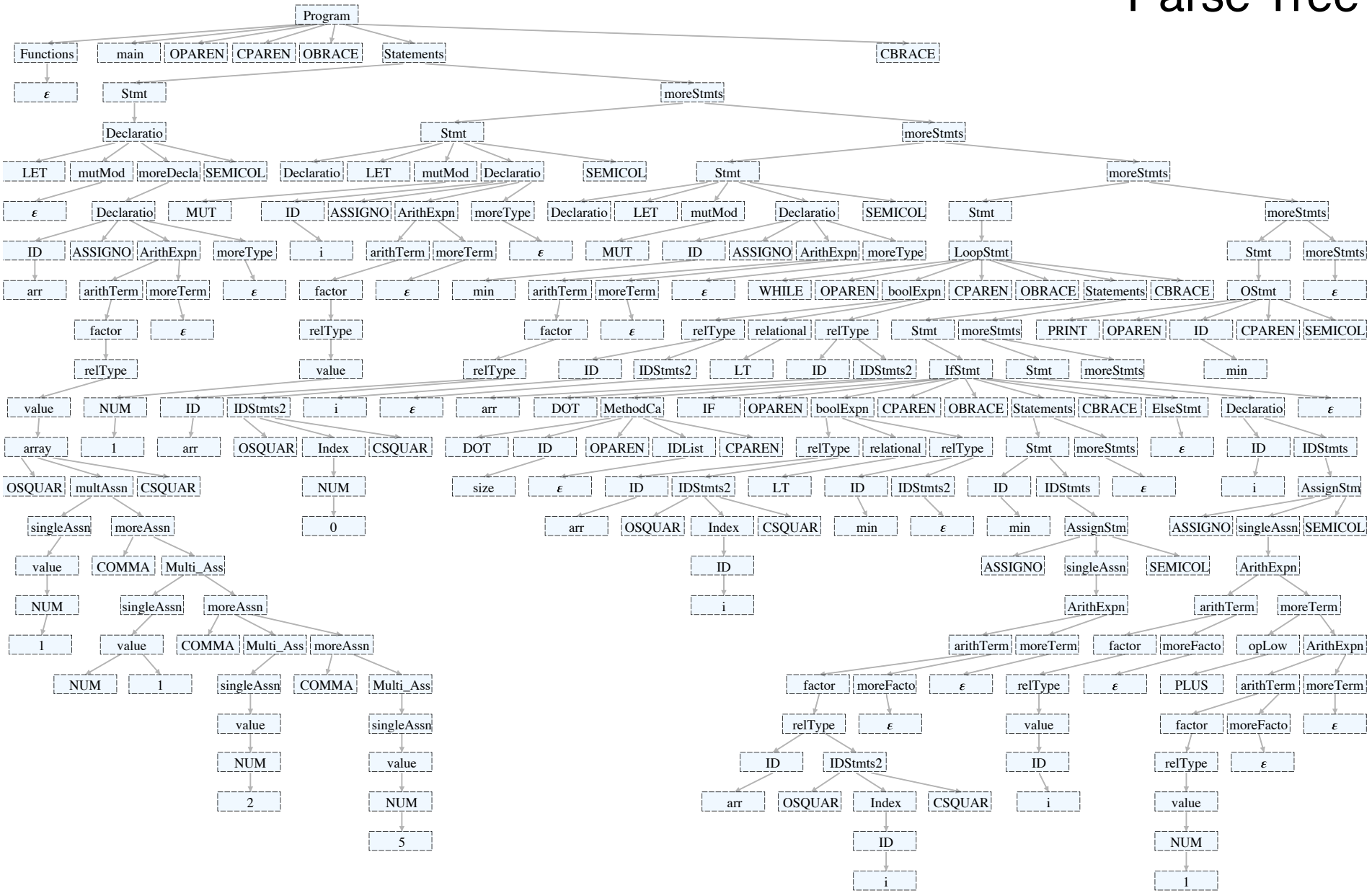
Parse Tree 2


Program
Functions
fn
main
{
Statements
}
ε
Stmt
more_Stmts
DeclarationStmt
Stmt
let
mut
Declaration
SEMICOLON
IfStmt
ElseStmt
ε
ID
ASSIGNOP
Arith_Expn
more_types
IF
(
boolExpn
)
{
Statements
}
ε
ArithTerm
more_terms
ε
relType
relationalOp
relType
Stmt
more_Stmts
factor
ε
ID
IDStmts2
==
ID
IDStmts2
OStmt
ε
relType
ε
MethodCall
print
SEMICOLON
value
DOT
ID
FunCall
(
ID
)
"string"
(
IDList
)
ε

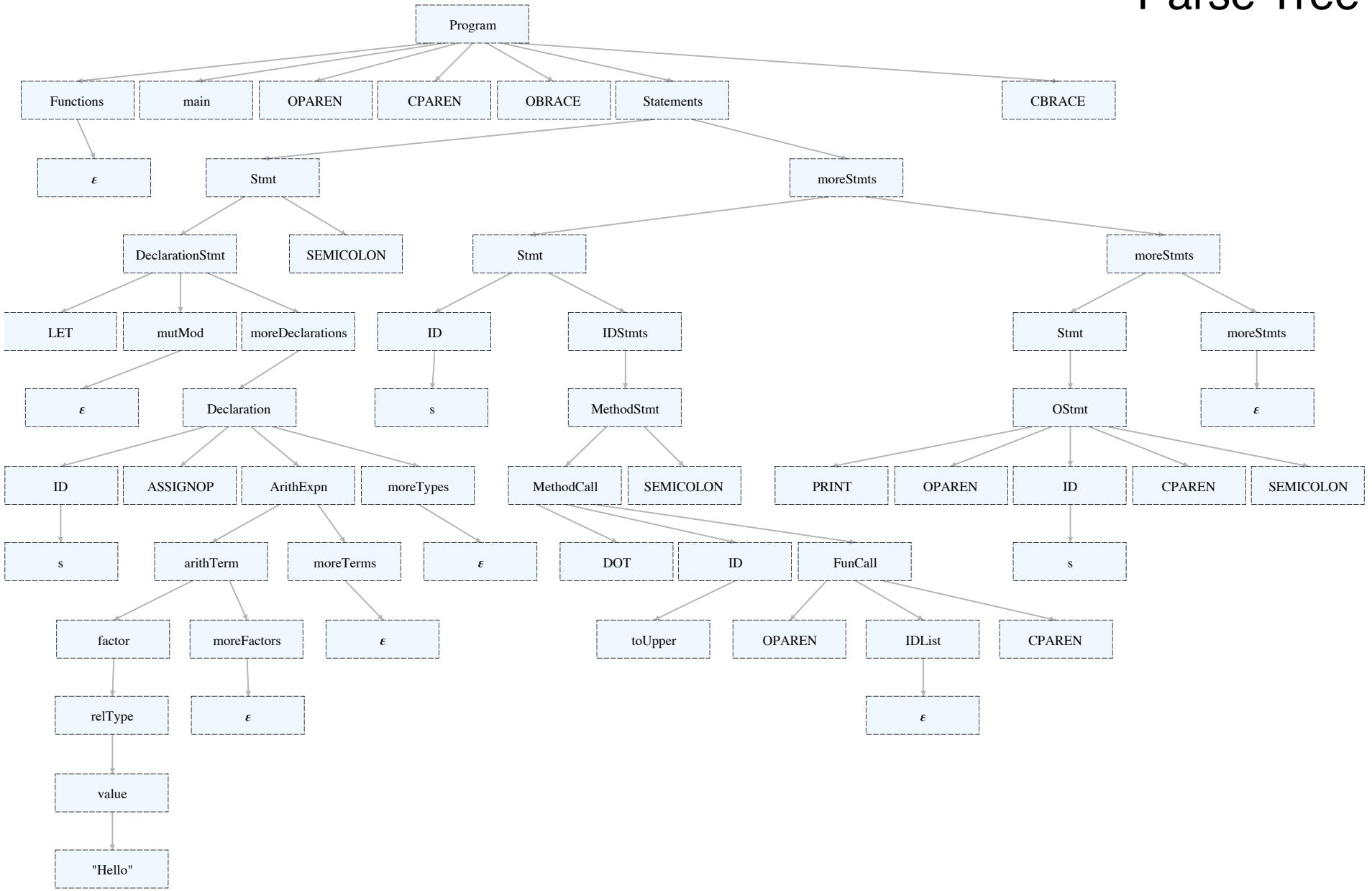
Parse Tree 3


Program
Functions
main
OPAREN
CPAREN
OBRACE
Statements
CBRACE
FunctionD
Functions
Stmt
moreStmts
FN
ID
OPAREN
parameter
CPAREN
fnReturn
OBRACE
Statements
CBRACE
ε
Declaratio
Stmt
moreStmts
ID
COLON
Type
remaining
->
Type
Stmt
LET
mutMod
Declaratio
SEMICOL
ID
IDStmts
Stmt
moreStmts
I32
COMMA
parameter
remaining
I32
Declaratio
moreStmts
ε
ID
ASSIGNO
Arith_Exp
moreTypes
AssignStm
OStmt
ε
ID
COLON
Type
ε
LET
mutMod
Declaratio
SEMICOL
Stmt
moreStmts
arithTerm
ε
ASSIGNO
singleAssn
PRINT
OPAREN
ID
CPAREN
SEMICOL
I32
ε
ID
ASSIGNO
Arith_Exp
moreTypes
ID
IDStmts
Stmt
moreStmts
factor
ArithExpn
arithTerm
ε
AssignStm
ReturnStm
ε
relType
arithTerm
factor
ASSIGNO
singleAssn
SEMICOL
RETURN
relType
SEMICOL
value
factor
relType
ArithExpn
ID
IDStmts2
NUM
relType
value
arithTerm
moreTerm
ε
ID
IDStmts2
NUM
factor
moreFacto
opLow
FunCall
ID
ε
ArithExpn
OPAREN
IDList
CPAREN
arithTerm
moreTerm
ID
moreIds
factor
moreFacto
ε
COMMA
IDList
ID
ε
ID
moreIds
ε

Parse Tree 4


Program
Functions
main
OPAREN
CPAREN
OBRACE
Statements
CBRACE
ε
Stmt
moreStmts
Declaratio
Stmt
moreStmts
LET
mutMod
moreDecla
SEMICOL
Declaratio
LET
mutMod
Declaratio
SEMICOL
Stmt
moreStmts
ε
Declaratio
MUT
ID
ASSIGNO
ArithExpn
moreType
Declaratio
LET
mutMod
Declaratio
SEMICOL
Stmt
moreStmts
ID
ASSIGNO
ArithExpn
moreType
i
arithTerm
moreTerm
ε
MUT
ID
ASSIGNO
ArithExpn
moreType
LoopStmt
Stmt
moreStmts
arr
arithTerm
moreTerm
ε
factor
ε
min
arithTerm
moreTerm
ε
WHILE
OPAREN
boolExpn
CPAREN
OBRACE
Statements
CBRACE
OStmt
ε
factor
ε
relType
factor
ε
relType
relational
relType
Stmt
moreStmts
PRINT
OPAREN
ID
CPAREN
SEMICOL
relType
value
relType
ID
IDStmts2
LT
ID
IDStmts2
IfStmt
Stmt
moreStmts
min
value
NUM
ID
IDStmts2
i
ε
arr
DOT
MethodCa
IF
OPAREN
boolExpn
CPAREN
OBRACE
Statements
CBRACE
ElseStmt
Declaratio
ε
array
1
arr
OSQUAR
Index
CSQUAR
DOT
ID
OPAREN
IDList
CPAREN
relType
relational
relType
Stmt
moreStmts
ε
ID
IDStmts
OSQUAR
multAssn
CSQUAR
NUM
size
ε
ID
IDStmts2
LT
ID
IDStmts2
ID
IDStmts
ε
i
AssignStm
singleAssn
moreAssn
0
arr
OSQUAR
Index
CSQUAR
min
ε
min
AssignStm
ASSIGNO
singleAssn
SEMICOL
value
COMMA
Multi_Ass
ID
ASSIGNO
singleAssn
SEMICOL
ArithExpn
NUM
singleAssn
moreAssn
i
ArithExpn
arithTerm
moreTerm
1
value
COMMA
Multi_Ass
moreAssn
arithTerm
moreTerm
factor
moreFacto
opLow
ArithExpn
NUM
1
singleAssn
COMMA
Multi_Ass
factor
moreFacto
ε
relType
ε
PLUS
arithTerm
moreTerm
value
singleAssn
relType
ε
value
factor
moreFacto
ε
NUM
2
value
ID
IDStmts2
ID
relType
ε
NUM
5
arr
OSQUAR
Index
CSQUAR
i
value
ID
i
NUM
1

Parse Tree 5


Program
Functions
main
OPAREN
CPAREN
OBRACE
Statements
CBRACE
ε
Stmt
moreStmts
DeclarationStmt
SEMICOLON
Stmt
moreStmts
LET
mutMod
moreDeclarations
ID
IDStmts
Stmt
moreStmts
ε
Declaration
s
MethodStmt
OStmt
ε
ID
ASSIGNOP
ArithExpn
moreTypes
MethodCall
SEMICOLON
PRINT
OPAREN
ID
CPAREN
SEMICOLON
s
arithTerm
moreTerms
ε
DOT
ID
FunCall
s
factor
moreFactors
ε
toUpper
OPAREN
IDList
CPAREN
relType
ε
ε
value
"Hello"