Team 10

Compiler Term-Project #2

The implementation of a bottom-up syntax analyzer

Date	June 27, 2020
Instructor	Hyosu Kim
Team	#10
Members	Heesang Ro (20145001) Junhyuck Woo (20145337)

INDEX

SPECIFICATIONS

CFG G:

1. CODE → VDECL CODE | FDECL CODE | ε
2. VDECL → vtype id semi | vtype ASSIGN semi
3. ASSIGN → id assign RHS
4. FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
5. ARG → vtype id MOREARGS | ε
6. MOREARGS → comma vtype id MOREARGS | ε
7. BLOCK → STMT BLOCK | ε
8. STMT → VDECL | ASSIGN semi
9. STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE
10. STMT → while lparen COND rparen lbrace BLOCK rbrace
11. STMT → for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace
12. ELSE → else lbrace BLOCK rbrace | ε
13. RHS → EXPR | literal
14. EXPR → TERM addsub EXPR | TERM
15. TERM → FACTOR multdiv TERM | FACTOR
16. FACTOR → lparen EXPR rparen | id | num | float
17. COND → FACTOR comp FACTOR
18. RETURN → return FACTOR semi

Terminals

1. **vtype** for the types of variables and functions
2. **num** for signed integers
3. **float** for floating-point numbers
4. **literal** for literal strings
5. **id** for the identifiers of variables and functions
6. **if**, **else**, **while**, **for** and **return** for if, else, while, for and return statements respectively
7. **addsub** for + and - arithmetic operators
8. **multdiv** for * and / arithmetic operators
9. **assign** for assignment operators
10. **comp** for comparison operators
11. **semi** and **comma** for semicolons and commas respectively
12. **lparen**, **rparen**, **lbrace**, and **rbrace** for (,), {, and } respectively

Non-terminals

CODE, VDECL, FDECL, ARG, MOREARGS, BLOCK, STMT, ASSIGN, RHS, EXPR, TERM, FACTOR, COND, RETURN, ELSE

Start symbol

CODE

Modified part in specifications

In the CFG, we add one more line.

00: S’ → CODE

FIRST SET

This is the result of First Set. You can check our handwriting version at the appendix 1.

First(RETURN) = {return}

First(FACTOR) = {lparen, id, num, float}

First(COND) = {lparen, id, num, float}

Fist(TERM) = {lparen, id, num, float}

First(EXPR) = {lparen, id, num, float}

First(RHS) = {lparen, id, num, float, literal}

First(ELSE) = {else, ε}

First (VDECL) = {vtype}

Fist (ASSIGN) = {id}

First (STMT) = {for, while, if, vtype, id}

First (BLOCK) = {for, while, if, vtype, id, ε}

First (MOREARGS) = {comma, ε}

First (ARG) = {vtype, ε}

First (FDECL) = {vtype}

First (CODE) = {vtype, ε}

First (S') = {vtype, ε}

FOLLOW SET

This is the result of Follow Set. You can check our handwriting version at the appendix 2.

Follow (S') = {$}

Follow (CODE) = {$}

Follow (VDECL) = {vtype, rbrace, return, for, while, if, id, $}

Follow (ASSIGN) = {semi, rparen}

Follow (FDECL) = {$, vtype}

Follow (ARG) = {rparen}

Follow (MOREARGS) = {rparen}

Follow (BLOCK) = {rbrace, return}

Follow (STMT) = {rbrace, return, if, while, for, vtype, id}

Follow (ELSE) = {rbrace, return, if, while, for, vtype, id}

Follow (RHS) = {semi, rparen}

Follow (EXPR) = {semi, rparen}

Follow (TERM) = {addsub, rparen, semi}

Follow (FACTOR) = {semi, comp, multdiv, rparen, addsub}

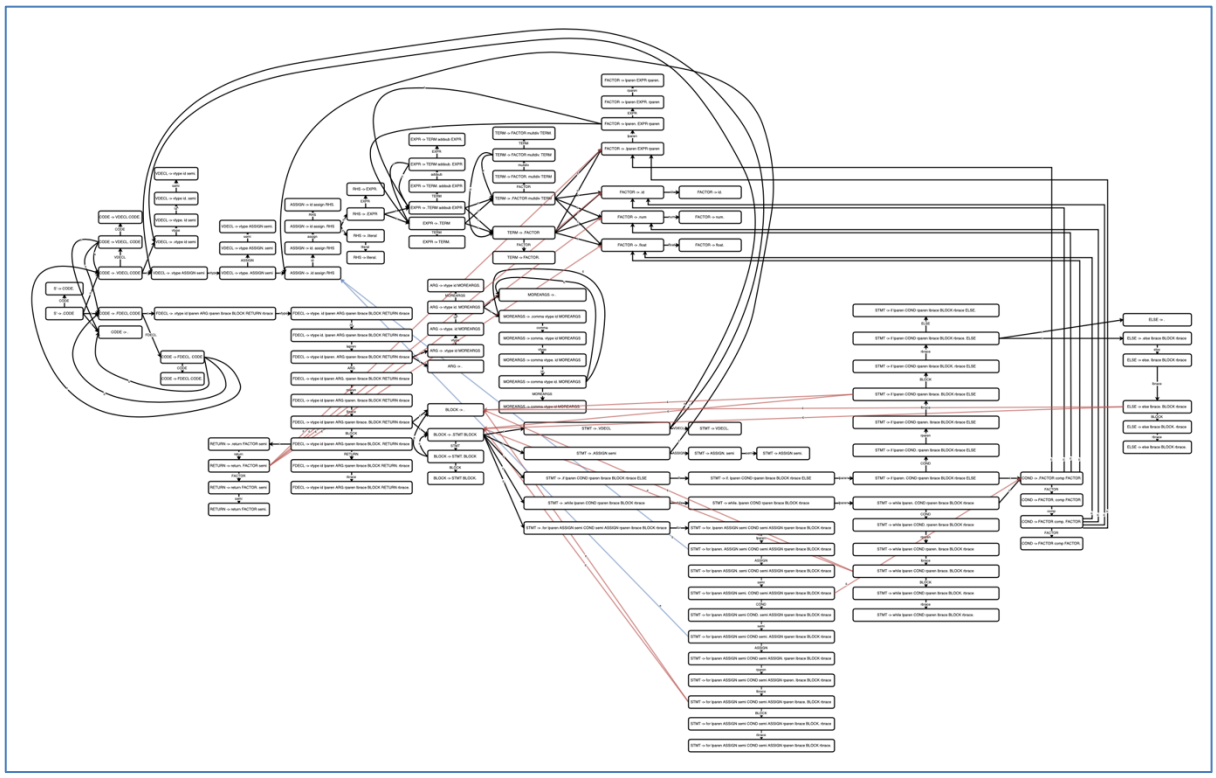
Follow (COND) = {semi, rparen}

Follow (RETURN) = {rbrace}

NFA (Non-deterministic Finite Automata)

Our team drew the NAF (Non-deterministic Finite Automata) using flow chart drawing tool[1]. The graph is too big, so it is hard to see the detail, so we included the drawing file in the “Handwriting” folder, if you want please check the directory.

1. Overall Graph



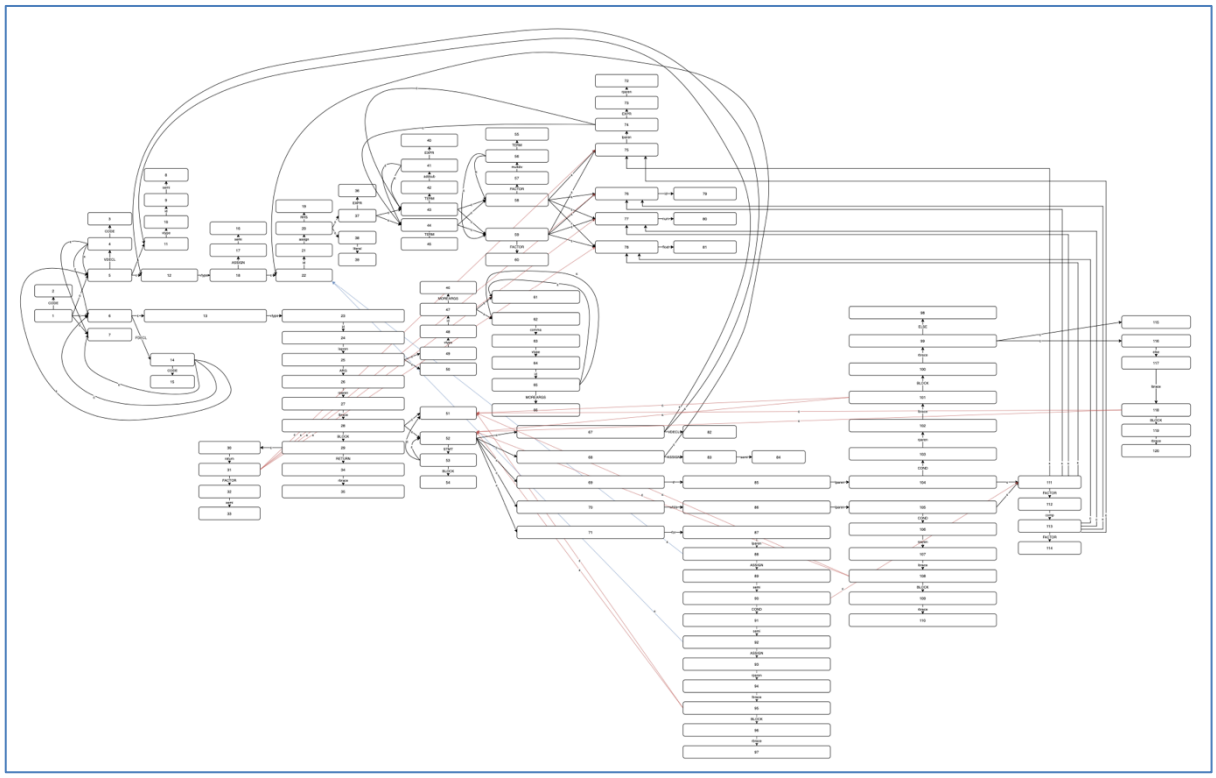
[1] https://app.diagrams.net

DFA (Deterministic Finite Automata)

DFA is generated using the NFA which we built. At first, we attached the number on each node and derived the DFA using subset (powerset) construction algorithm. You can check the handwriting version at the appendix 3.

1. NFA with numbering

This image also can be found at the “Handwriting” directory.



2. Subset Construction

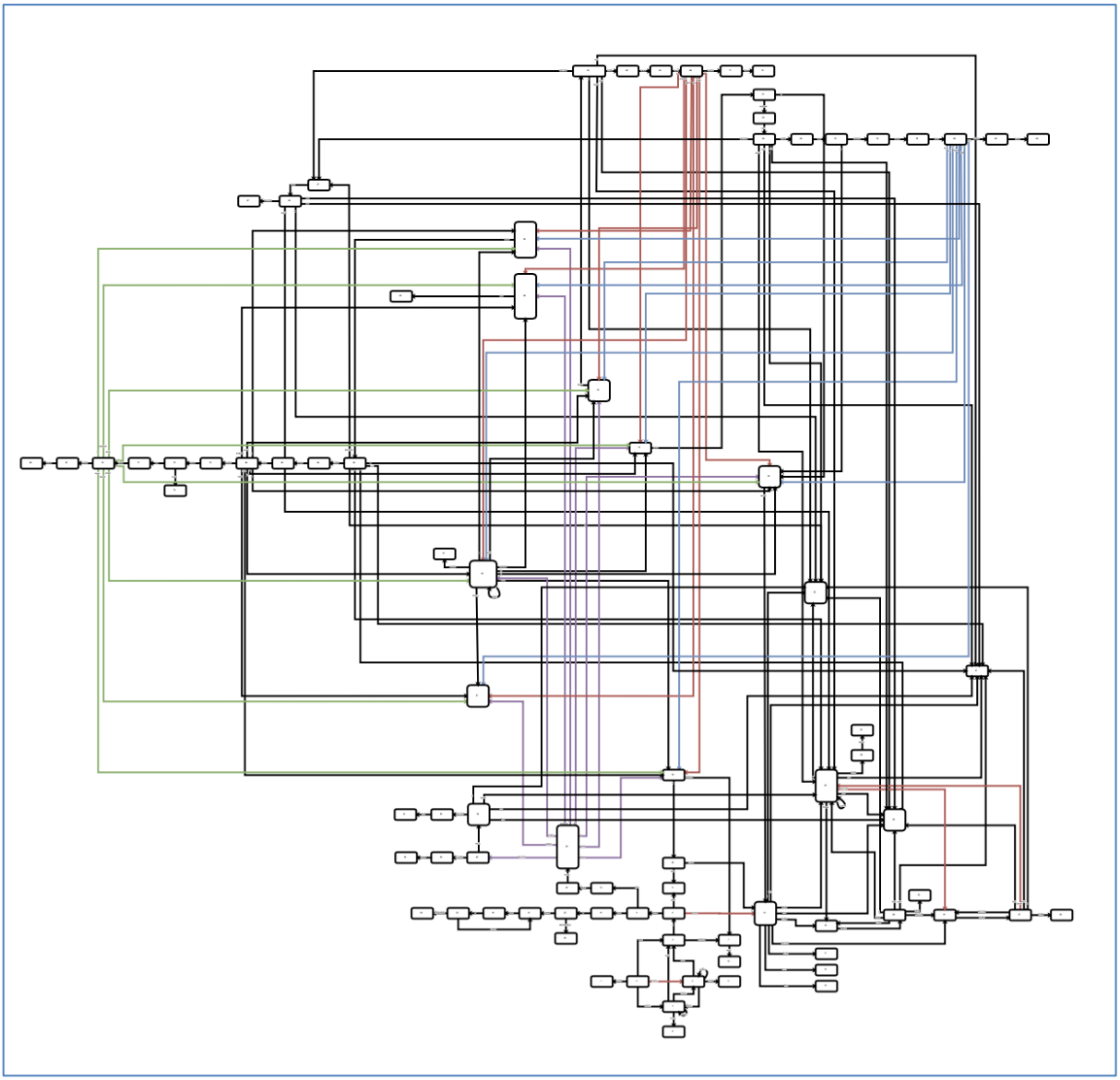
$\varepsilon - closure(1) = T_0$
$\varepsilon - closure(\delta(T_0,\ CODE)) = T_1$
$\varepsilon - closure(\delta(T_0, FDECL)) = T_2$
$\varepsilon - closure(\delta(T_0, vtype)) = T_3$
$\varepsilon - closure(\delta(T_0, VDECL)) = T_4$
$\varepsilon - closure(\delta(T_2, CODE)) = T_5$
$\varepsilon - closure(\delta(T_2, FDECL)) = T_2$
$\varepsilon - closure(\delta(T_2, vtype)) = T_3$
$\varepsilon - closure(\delta(T_2, VDECL)) = T_4$
$\varepsilon - closure(\delta(T_3, id)) = T_6$
$\varepsilon - closure(\delta(T_3, ASSIGN)) = T_7$
$\varepsilon - closure(\delta(T_4, CODE)) = T_8$
$\varepsilon - closure(\delta(T_4, VDECL)) = T_4$
$\varepsilon - closure(\delta(T_4, vtype)) = T_3$
$\varepsilon - closure(\delta(T_4, FDECL)) = T_2$
$\varepsilon - closure(\delta(T_6, lparen)) = T_9$
$\varepsilon - closure(\delta(T_6, assign)) = T_{10}$
$\varepsilon - closure(\delta(T_6, semi)) = T_{11}$
$\varepsilon - closure(\delta(T_7, semi)) = T_{12}$
$\varepsilon - closure(\delta(T_9, ARG)) = T_{13}$
$\varepsilon - closure(\delta(T_9, vtype)) = T_{14}$
$\varepsilon - closure(\delta(T_{10}, RHS)) = T_{15}$
$\varepsilon - closure(\delta(T_{10}, literal)) = T_{16}$
$\varepsilon - closure(\delta(T_{1o}, EXPR)) = T_{17}$
$\varepsilon - closure(\delta(T_{10}, TERM)) = T_{18}$
$\varepsilon - closure(\delta(T_{10}, FACTOR)) = T_{19}$
$\varepsilon - closure(\delta(T_{10}, float)) = T_{20}$
$\varepsilon - closure(\delta(T_{10}, num)) = T_{21}$
$\varepsilon - closure(\delta(T_{10}, id)) = T_{22}$
$\varepsilon - closure(\delta(T_{10}, lparen)) = T_{23}$
$\varepsilon - closure(\delta(T_{13}, rparen)) = T_{24}$
$\varepsilon - closure(\delta(T_{14}, id)) = T_{25}$
$\varepsilon - closure(\delta(T_{18}, addsub)) = T_{26}$
$\varepsilon - closure(\delta(T_{19}, multdiv)) = T_{27}$
$\varepsilon - closure(\delta(T_{23}, EXPR)) = T_{28}$
$\varepsilon - closure(\delta(T_{23}, TERM)) = T_{18}$
$\varepsilon - closure(\delta(T_{23}, FACTOR)) = T_{19}$
$\varepsilon - closure(\delta(T_{23}, float)) = T_{20}$
$\varepsilon - closure(\delta(T_{23}, num)) = T_{21}$
$\varepsilon - closure(\delta(T_{23}, id)) = T_{22}$
$\varepsilon - closure(\delta(T_{23}, lparen)) = T_{23}$
$\varepsilon - closure(\delta(T_{24}, lparen)) = T_{29}$
$\varepsilon - closure(\delta(T_{25}, MOREARGS)) = T_{30}$
$\varepsilon - closure(\delta(T_{25}, comma)) = T_{31}$
$\varepsilon - closure(\delta(T_{26}, EXPR)) = T_{32}$
$\varepsilon - closure(\delta(T_{26}, TERM)) = T_{18}$
$\varepsilon - closure(\delta(T_{26}, FACTOR)) = T_{19}$
$\varepsilon - closure(\delta(T_{26}, float)) = T_{20}$
$\varepsilon - closure(\delta(T_{26}, num)) = T_{21}$
$\varepsilon - closure(\delta(T_{26}, id)) = T_{22}$
$\varepsilon - closure(\delta(T_{26}, lparen)) = T_{23}$
$\varepsilon - closure(\delta(T_{27}, TERM)) = T_{33}$
$\varepsilon - closure(\delta(T_{27}, FACTOR)) = T_{19}$
$\varepsilon - closure(\delta(T_{27}, float)) = T_{20}$
$\varepsilon - closure(\delta(T_{27}, num)) = T_{21}$
$\varepsilon - closure(\delta(T_{27}, id)) = T_{22}$
$\varepsilon - closure(\delta(T_{27}, lparen)) = T_{23}$
$\varepsilon - closure(\delta(T_{28}, rparen)) = T_{34}$
$\varepsilon - closure(\delta(T_{29}, BLOCK)) = T_{35}$
$\varepsilon - closure(\delta(T_{29}, STMT)) = T_{36}$
$\varepsilon - closure(\delta(T_{29}, VDECL)) = T_{37}$
$\varepsilon - closure(\delta(T_{29}, ASSIGN)) = T_{38}$
$\varepsilon - closure(\delta(T_{29}, if)) = T_{39}$
$\varepsilon - closure(\delta(T_{29}, while)) = T_{40}$
$\varepsilon - closure(\delta(T_{29}, for)) = T_{41}$
$\varepsilon - closure(\delta(T_{29}, vtype)) = T_{42}$
$\varepsilon - closure(\delta(T_{29}, id)) = T_{43}$
$\varepsilon - closure(\delta(T_{31}, vtype)) = T_{44}$
$\varepsilon - closure(\delta(T_{35}, RETURN)) = T_{45}$
$\varepsilon - closure(\delta(T_{35}, return)) = T_{46}$
$\varepsilon - closure(\delta(T_{36}, BLOCK)) = T_{47}$
$\varepsilon - closure(\delta(T_{36}, STMT)) = T_{36}$
$\varepsilon - closure(\delta(T_{36}, VDECL)) = T_{37}$
$\varepsilon - closure(\delta(T_{36}, ASSIGN)) = T_{38}$
$\varepsilon - closure(\delta(T_{36}, if)) = T_{39}$
$\varepsilon - closure(\delta(T_{36}, while)) = T_{40}$
$\varepsilon - closure(\delta(T_{36}, for)) = T_{41}$
$\varepsilon - closure(\delta(T_{36}, vtype)) = T_{42}$
$\varepsilon - closure(\delta(T_{36}, id)) = T_{43}$
$\varepsilon - closure(\delta(T_{38}, semi)) = T_{48}$
$\varepsilon - closure(\delta(T_{39}, lparen)) = T_{49}$
$\varepsilon - closure(\delta(T_{40}, lparen)) = T_{50}$
$\varepsilon - closure(\delta(T_{41}, lparen)) = T_{51}$
$\varepsilon - closure(\delta(T_{42}, id)) = T_{52}$
$\varepsilon - closure(\delta(T_{42}, ASSIGN)) = T_7$
$\varepsilon - closure(\delta(T_{43}, assign)) = T_{10}$
$\varepsilon - closure(\delta(T_{44}, id)) = T_{53}$
$\varepsilon - closure(\delta(T_{45}, rbrace)) = T_{54}$
$\varepsilon - closure(\delta(T_{46}, FACTOR)) = T_{55}$
$\varepsilon - closure(\delta(T_{46}, float)) = T_{20}$
$\varepsilon - closure(\delta(T_{46}, num)) = T_{21}$
$\varepsilon - closure(\delta(T_{46}, id)) = T_{22}$
$\varepsilon - closure(\delta(T_{46}, lparen)) = T_{23}$
$\varepsilon - closure(\delta(T_{49}, COND)) = T_{56}$
$\varepsilon - closure(\delta(T_{49}, FACTOR)) = T_{57}$
$\varepsilon - closure(\delta(T_{49}, float)) = T_{20}$
$\varepsilon - closure(\delta(T_{49}, num)) = T_{21}$
$\varepsilon - closure(\delta(T_{49}, id)) = T_{22}$

$\varepsilon - closure(\delta(T_{49}, lparen))$ = T_{23}
$\varepsilon - closure(\delta(T_{50}, COND))$ = T_{58}
$\varepsilon - closure(\delta(T_{50}, FACTOR))$ = T_{57}
$\varepsilon - closure(\delta(T_{50}, float))$ = T_{20}
$\varepsilon - closure(\delta(T_{50}, num))$ = T_{21}
$\varepsilon - closure(\delta(T_{50}, id))$ = T_{22}
$\varepsilon - closure(\delta(T_{50}, lparen))$ = T_{23}
$\varepsilon - closure(\delta(T_{51}, ASSIGN))$ = T_{59}
$\varepsilon - closure(\delta(T_{51}, id))$ = T_{43}
$\varepsilon - closure(\delta(T_{52}, semi))$ = T_{11}
$\varepsilon - closure(\delta(T_{52}, assign))$ = T_{10}
$\varepsilon - closure(\delta(T_{53}, MOREARGS))$ = T_{60}
$\varepsilon - closure(\delta(T_{53}, comma))$ = T_{31}
$\varepsilon - closure(\delta(T_{55}, semi))$ = T_{61}
$\varepsilon - closure(\delta(T_{56}, rparen))$ = T_{62}
$\varepsilon - closure(\delta(T_{57}, comp))$ = T_{63}
$\varepsilon - closure(\delta(T_{58}, rparen))$ = T_{64}
$\varepsilon - closure(\delta(T_{59}, semi))$ = T_{65}
$\varepsilon - closure(\delta(T_{62}, lbrace))$ = T_{66}
$\varepsilon - closure(\delta(T_{63}, FACTOR))$ = T_{67}
$\varepsilon - closure(\delta(T_{63}, float))$ = T_{20}
$\varepsilon - closure(\delta(T_{63}, num))$ = T_{21}
$\varepsilon - closure(\delta(T_{63}, id))$ = T_{22}
$\varepsilon - closure(\delta(T_{63}, lparen))$ = T_{23}
$\varepsilon - closure(\delta(T_{64}, lbrace))$ = T_{68}
$\varepsilon - closure(\delta(T_{65}, COND))$ = T_{69}
$\varepsilon - closure(\delta(T_{65}, FACTOR))$ = T_{57}
$\varepsilon - closure(\delta(T_{65}, float))$ = T_{20}
$\varepsilon - closure(\delta(T_{65}, num))$ = T_{21}
$\varepsilon - closure(\delta(T_{65}, id))$ = T_{22}
$\varepsilon - closure(\delta(T_{65}, lparen))$ = T_{23}
$\varepsilon - closure(\delta(T_{66}, BLOCK))$ = T_{70}
$\varepsilon - closure(\delta(T_{66}, STMT))$ = T_{36}
$\varepsilon - closure(\delta(T_{66}, VDECL))$ = T_{37}
$\varepsilon - closure(\delta(T_{66}, ASSIGN))$ = T_{38}
$\varepsilon - closure(\delta(T_{66}, if))$ = T_{39}
$\varepsilon - closure(\delta(T_{66}, while))$ = T_{40}
$\varepsilon - closure(\delta(T_{66}, for))$ = T_{41}
$\varepsilon - closure(\delta(T_{66}, vtype))$ = T_{42}
$\varepsilon - closure(\delta(T_{66}, id))$ = T_{43}
$\varepsilon - closure(\delta(T_{68}, BLOCK))$ = T_{71}
$\varepsilon - closure(\delta(T_{68}, STMT))$ = T_{36}
$\varepsilon - closure(\delta(T_{68}, VDECL))$ = T_{37}
$\varepsilon - closure(\delta(T_{68}, ASSIGN))$ = T_{38}
$\varepsilon - closure(\delta(T_{68}, if))$ = T_{39}
$\varepsilon - closure(\delta(T_{68}, while))$ = T_{40}
$\varepsilon - closure(\delta(T_{68}, for))$ = T_{41}
$\varepsilon - closure(\delta(T_{68}, vtype))$ = T_{42}
$\varepsilon - closure(\delta(T_{68}, id))$ = T_{43}
$\varepsilon - closure(\delta(T_{69}, semi))$ = T_{72}
$\varepsilon - closure(\delta(T_{70}, rbrace))$ = T_{73}
$\varepsilon - closure(\delta(T_{71}, rbrace))$ = T_{74}
$\varepsilon - closure(\delta(T_{72}, ASSIGN))$ = T_{75}
$\varepsilon - closure(\delta(T_{72}, id))$ = T_{43}
$\varepsilon - closure(\delta(T_{73}, ELSE))$ = T_{76}
$\varepsilon - closure(\delta(T_{73}, else))$ = T_{77}
$\varepsilon - closure(\delta(T_{75}, rparen))$ = T_{78}
$\varepsilon - closure(\delta(T_{77}, lbrace))$ = T_{79}
$\varepsilon - closure(\delta(T_{78}, lbrace))$ = T_{80}
$\varepsilon - closure(\delta(T_{79}, BLOCK))$ = T_{81}
$\varepsilon - closure(\delta(T_{79}, STMT))$ = T_{36}
$\varepsilon - closure(\delta(T_{79}, VDECL))$ = T_{37}
$\varepsilon - closure(\delta(T_{79}, ASSIGN))$ = T_{38}
$\varepsilon - closure(\delta(T_{79}, if))$ = T_{39}
$\varepsilon - closure(\delta(T_{79}, while))$ = T_{40}
$\varepsilon - closure(\delta(T_{79}, for))$ = T_{41}
$\varepsilon - closure(\delta(T_{79}, vtype))$ = T_{42}
$\varepsilon - closure(\delta(T_{79}, id))$ = T_{43}
$\varepsilon - closure(\delta(T_{80}, BLOCK))$ = T_{82}
$\varepsilon - closure(\delta(T_{80}, STMT))$ = T_{36}
$\varepsilon - closure(\delta(T_{80}, VDECL))$ = T_{37}
$\varepsilon - closure(\delta(T_{80}, ASSIGN))$ = T_{38}
$\varepsilon - closure(\delta(T_{80}, if))$ = T_{39}
$\varepsilon - closure(\delta(T_{80}, while))$ = T_{40}
$\varepsilon - closure(\delta(T_{80}, for))$ = T_{41}
$\varepsilon - closure(\delta(T_{80}, vtype))$ = T_{42}
$\varepsilon - closure(\delta(T_{80}, id))$ = T_{43}
$\varepsilon - closure(\delta(T_{81}, rbrace))$ = T_{83}
$\varepsilon - closure(\delta(T_{82}, rbrace))$ = T_{84}

3. Graph

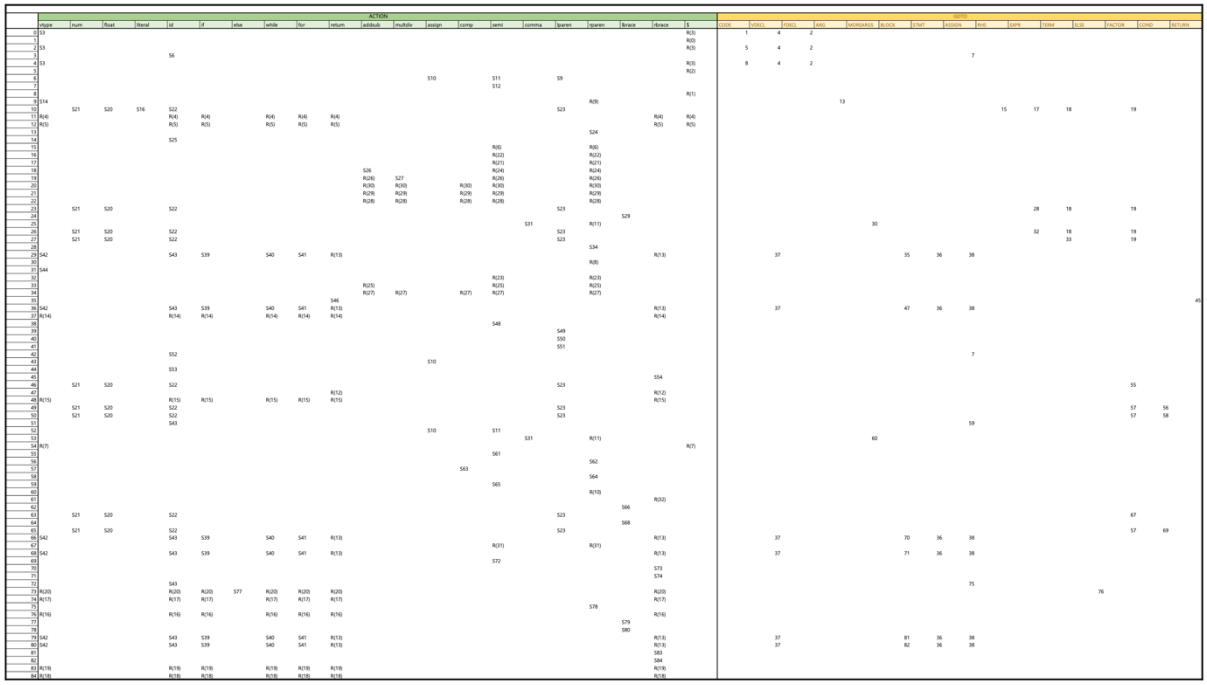
This image also can be found at the “Handwriting” directory.

To generate the parsing table, graph node should include the CFG. However, there are too many things to include at the single node, so instead to do that we make the reference list. You can check the list at appendix 4.



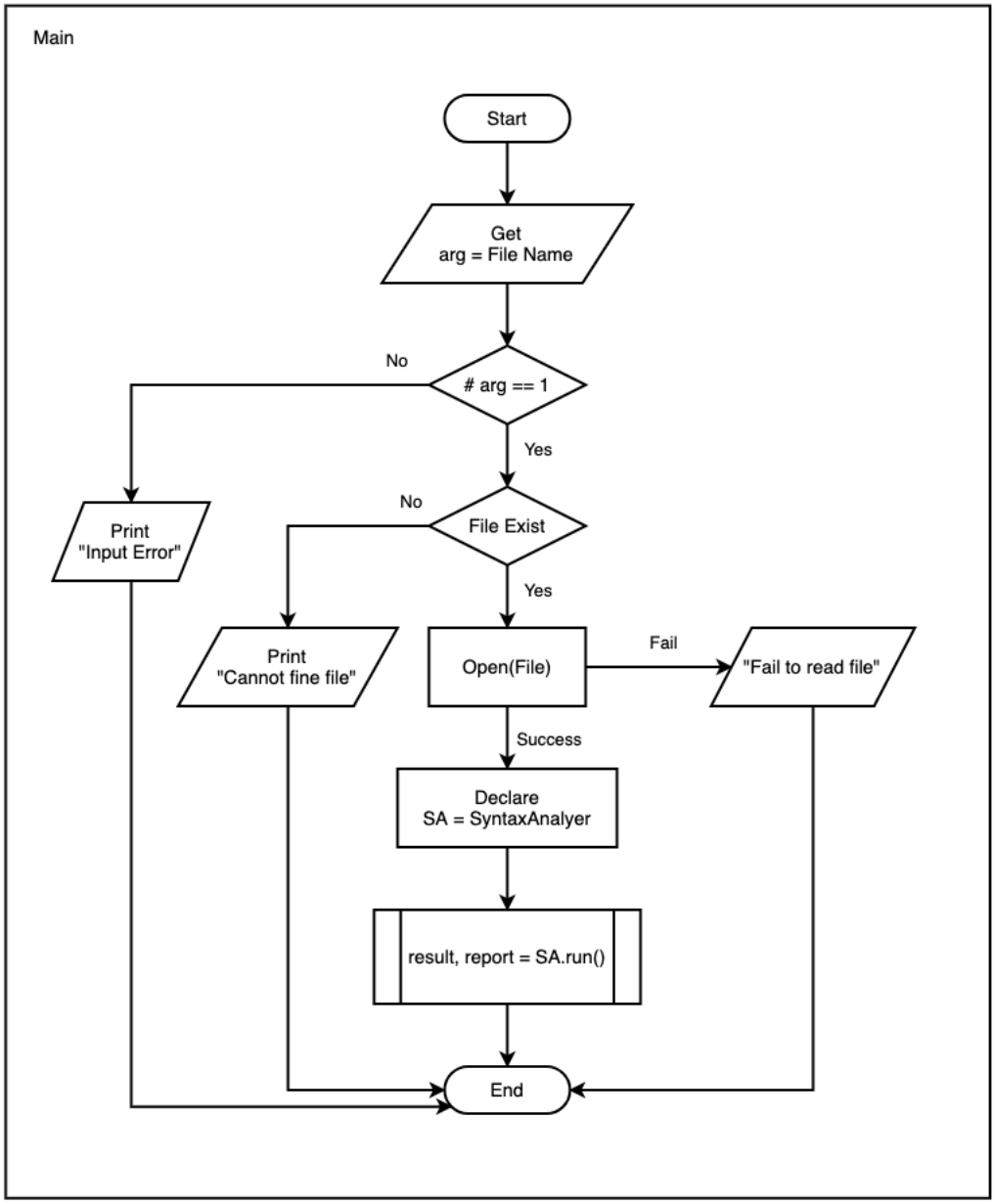
SLR PARSING TABLE

The number of terminals is 20, the number of non-terminal is 15, and the number of DFA node is 85. It means that the matrix is huge. When filling in the “reduce” operation, we refer to the rules and follow. You already read the follow set in appendix 2, and you can see the rules in appendix 5. The table is huge to render at the limited space, so we include the SLR parsing table at the “Handwriting” directory.

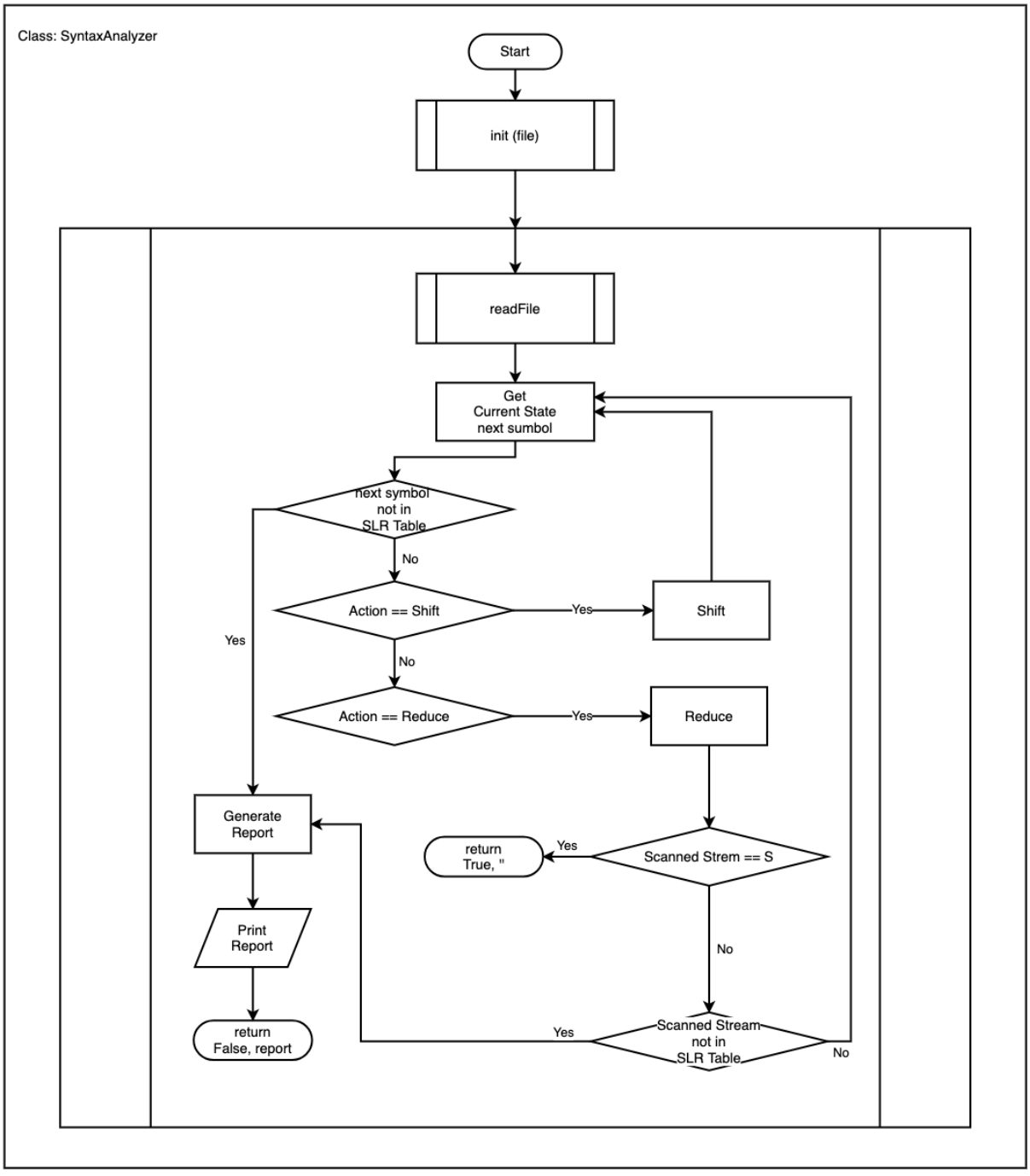


CODE ALGORITHM

- ✓ Main



✓ Class SyntaxAnalyzer



IMPLEMENTATION

Before explaining our works, we introduce the developing environment.

Language
- ✓ Python3 (version: 3.7.4)

Operating System
- ✓ macOS Catalina
- ✓ Windows 10

IDE (Integrated Development Environment)
- ✓ Visual Studio Code (version: 1.45.0)
- ✓ PyCharm (version: 3.9.4)

Project Management
- ✓ Git (version: 2.24.2)
- ✓ Git-Hub

1. Definition of Rules

When calculating the reduce operation, we need to refer to the rules. And there is a relationship between numbers from reduce process and rules. To use this relationship, we implement this using the dictionary structure.

```
RULES = {'0':'S → CODE',
         '1':'CODE → VDECL CODE',
         '2':'CODE → FDECL CODE',
         '3':'CODE → epsilon',
         '4':'VDECL → vtype id semi',
         '5':'VDECL → vtype ASSIGN semi',
         '6':'ASSIGN → id assign RHS',
         '7':'FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace',
         '8':'ARG → vtype id MOREARGS',
         '9':'ARG → epsilon',
         '10':'MOREARGS → comma vtype id MOREARGS',
         '11':'MOREARGS → epsilon',
         '12':'BLOCK → STMT BLOCK',
         '13':'BLOCK → epsilon',
         '14':'STMT → VDECL',
         '15':'STMT → ASSIGN semi',
         '16':'STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE',
         '17':'STMT → while lparen COND rparen lbrace BLOCK rbrace',
         '18':'STMT → for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace',
         '19':'ELSE → else lbrace BLOCK rbrace',
         '20':'ELSE → epsilon',
         '21':'RHS → EXPR',
         '22':'RHS → literal',
         '23':'EXPR → TERM addsub EXPR',
         '24':'EXPR → TERM',
         '25':'TERM → FACTOR multdiv TERM',
         '26':'TERM → FACTOR',
         '27':'FACTOR → lparen EXPR rparen',
         '28':'FACTOR → id',
         '29':'FACTOR → num',
         '30':'FACTOR → float',
         '31':'COND → FACTOR comp FACTOR',
         '32':'RETURN → return FACTOR semi'}
```

2. Definition of SLR Table

The SLR table has a lot of empty space. Such a matrix is called a sparse matrix. To save memory waste and better readability, we decide to use a list and dictionary. It is the structure in which the list holds a dictionary. A mechanism is simple, using the current state number, get the dictionary, and refer the operation using the next symbol as key.

```
SLR_TABLE = [{'vtype': 'S3', '$': 'R(3)', 'CODE': 1, 'VDECL': 4, 'FDECL': 2},
             {'$': 'R(0)'},
             {'vtype': 'S3', '$': 'R(3)', 'CODE': 5, 'VDECL': 4, 'FDECL': 2},
             {'id': 'S6', 'ASSIGN': 7},
             {'vtype': 'S3', '$': 'R(3)', 'CODE': 8, 'VDECL': 4, 'FDECL': 2},
             {'$': 'R(2)'},
             {'assign': 'S10', 'semi': 'S11', 'lparen': 'S9'},
             {'semi': 'S12'},
             {'$': 'R(1)'},
             {'vtype': 'S14', 'rparen': 'R(9)', 'ARG': 13},
             {'num': 'S21', 'float': 'S20', 'literal': 'S16', 'id': 'S22', 'lparen': 'S23', 'RHS': 15, 'EXPR': 17, 'TERM': 18, 'FACTOR': 19},
             {'vtype': 'R(4)', 'id': 'R(4)', 'if': 'R(4)', 'while': 'R(4)', 'for': 'R(4)', 'return': 'R(4)', 'rbrace': 'R(4)', '$': 'R(4)'},
             {'vtype': 'R(5)', 'id': 'R(5)', 'if': 'R(5)', 'while': 'R(5)', 'for': 'R(5)', 'return': 'R(5)', 'rbrace': 'R(5)', '$': 'R(5)'},
             {'rparen': 'S24'},
             {'id': 'S25'},
             {'semi': 'R(6)', 'rparen': 'R(6)'},
             {'semi': 'R(22)', 'rparen': 'R(22)'},
             {'semi': 'R(21)', 'rparen': 'R(21)'},
             {'addsub': 'S26', 'semi': 'R(24)', 'rparen': 'R(24)'},
             {'addsub': 'R(26)', 'multdiv': 'S27', 'semi': 'R(26)', 'rparen': 'R(26)'},
             {'addsub': 'R(30)', 'multdiv': 'R(30)', 'comp': 'R(30)', 'semi': 'R(30)', 'rparen': 'R(30)'},
             {'addsub': 'R(29)', 'multdiv': 'R(29)', 'comp': 'R(29)', 'semi': 'R(29)', 'rparen': 'R(29)'},
             {'addsub': 'R(28)', 'multdiv': 'R(28)', 'comp': 'R(28)', 'semi': 'R(28)', 'rparen': 'R(28)'},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'EXPR': 28, 'TERM': 18, 'FACTOR': 19},
             {'lbrace': 'S29'},
             {'comma': 'S31', 'rparen': 'R(11)', 'MOREARGS': 30},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'EXPR': 32, 'TERM': 18, 'FACTOR': 19},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'TERM': 33, 'FACTOR': 19},
             {'rparen': 'S34'},
             {'vtype': 'S42', 'id': 'S43', 'if': 'S39', 'while': 'S40', 'for': 'S41', 'return': 'R(13)', 'rbrace': 'R(13)', 'VDECL': 37, 'BLOCK': 35, 'STMT': 36, 'ASSIGN': 38},
             {'rparen': 'R(8)'},
             {'vtype': 'S44'},
             {'semi': 'R(23)', 'rparen': 'R(23)'},
             {'addsub': 'R(25)', 'semi': 'R(25)', 'rparen': 'R(25)'},
             {'addsub': 'R(27)', 'multdiv': 'R(27)', 'comp': 'R(27)', 'semi': 'R(27)', 'rparen': 'R(27)'},
             {'return': 'S46', 'RETURN': 45},
             {'vtype': 'S42', 'id': 'S43', 'if': 'S39', 'while': 'S40', 'for': 'S41', 'return': 'R(13)', 'rbrace': 'R(13)', 'VDECL': 37, 'BLOCK': 47, 'STMT': 36, 'ASSIGN': 38},
             {'vtype': 'R(14)', 'id': 'R(14)', 'if': 'R(14)', 'while': 'R(14)', 'for': 'R(14)', 'return': 'R(14)', 'rbrace': 'R(14)'},
             {'semi': 'S48'},
             {'lparen': 'S49'},
             {'lparen': 'S50'},
             {'lparen': 'S51'},
             {'id': 'S52', 'ASSIGN': 7},
             {'assign': 'S10'},
             {'id': 'S53'},
             {'rbrace': 'S54'},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'FACTOR': 55},
             {'return': 'R(12)', 'rbrace': 'R(12)'},
             {'vtype': 'R(15)', 'id': 'R(15)', 'if': 'R(15)', 'while': 'R(15)', 'for': 'R(15)', 'return': 'R(15)', 'rbrace': 'R(15)'},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'FACTOR': 57, 'COND': 56},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'FACTOR': 57, 'COND': 58},
             {'id': 'S43', 'ASSIGN': 59},
             {'assign': 'S10', 'semi': 'S11'},
             {'comma': 'S31', 'rparen': 'R(11)', 'MOREARGS': 60},
             {'vtype': 'R(7)', '$': 'R(7)'},
             {'semi': 'S61'},
             {'rparen': 'S62'},
             {'comp': 'S63'},
             {'rparen': 'S64'},
             {'semi': 'S65'},
             {'rparen': 'R(10)'},
             {'rbrace': 'R(32)'},
             {'lbrace': 'S66'},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'FACTOR': 67},
             {'lbrace': 'S68'},
             {'num': 'S21', 'float': 'S20', 'id': 'S22', 'lparen': 'S23', 'FACTOR': 57, 'COND': 69},
             {'vtype': 'S42', 'id': 'S43', 'if': 'S39', 'while': 'S40', 'for': 'S41', 'return': 'R(13)', 'rbrace': 'R(13)', 'VDECL': 37, 'BLOCK': 70, 'STMT': 36, 'ASSIGN': 38},
             {'semi': 'R(31)', 'rparen': 'R(31)'},
             {'vtype': 'S42', 'id': 'S43', 'if': 'S39', 'while': 'S40', 'for': 'S41', 'return': 'R(13)', 'rbrace': 'R(13)', 'VDECL': 37, 'BLOCK': 71, 'STMT': 36, 'ASSIGN': 38},
             {'semi': 'S72'},
             {'rbrace': 'S73'},
             {'rbrace': 'S74'},
             {'id': 'S43', 'ASSIGN': 75},
             {'vtype': 'R(20)', 'id': 'R(20)', 'if': 'R(20)', 'else': 'S77', 'while': 'R(20)', 'for': 'R(20)', 'return': 'R(20)', 'rbrace': 'R(20)', 'ELSE': 76},
             {'vtype': 'R(17)', 'id': 'R(17)', 'if': 'R(17)', 'while': 'R(17)', 'for': 'R(17)', 'return': 'R(17)', 'rbrace': 'R(17)'},
             {'rparen': 'S78'},
             {'vtype': 'R(16)', 'id': 'R(16)', 'if': 'R(16)', 'while': 'R(16)', 'for': 'R(16)', 'return': 'R(16)', 'rbrace': 'R(16)'},
             {'lbrace': 'S79'},
             {'lbrace': 'S80'},
             {'vtype': 'S42', 'id': 'S43', 'if': 'S39', 'while': 'S40', 'for': 'S41', 'return': 'R(13)', 'rbrace': 'R(13)', 'VDECL': 37, 'BLOCK': 81, 'STMT': 36, 'ASSIGN': 38},
             {'vtype': 'S42', 'id': 'S43', 'if': 'S39', 'while': 'S40', 'for': 'S41', 'return': 'R(13)', 'rbrace': 'R(13)', 'VDECL': 37, 'BLOCK': 82, 'STMT': 36, 'ASSIGN': 38},
             {'rbrace': 'S83'},
             {'rbrace': 'S84'},
             {'vtype': 'R(19)', 'id': 'R(19)', 'if': 'R(19)', 'while': 'R(19)', 'for': 'R(19)', 'return': 'R(19)', 'rbrace': 'R(19)'},
             {'vtype': 'R(18)', 'id': 'R(18)', 'if': 'R(18)', 'while': 'R(18)', 'for': 'R(18)', 'return': 'R(18)', 'rbrace': 'R(18)'}]
```

3. Splitter and Stack

We need to decide position of splitter and define state stack. Stack only has 0, position of splitter is 0 at first.

```
def run(self):
    # Read file
    self.readFile()
    #only includes end mark
    if (len(self.terminal_list)==1):
        return True, ''

    SLR_stack = [0]          #stack
    spliter_pos = 0  #position of spliter
    error_line = 1
```

Current state is decided by top of the stack. And next input symbol is decided by position of splitter.

```
#current state
current_state = SLR_stack[-1]

#next input symbol is deicded by position of spliter
next_input_symbol = self.terminal_list[spliter_pos]
```

4. Shift and Reduce

Shift is occurred when the value of SLR TABLE is like S73. It means shift and GOTO 73. 73 is added in our stack and splitter moves forward.

```
#shift
if (self.SLR_TABLE[current_state][next_input_symbol][0]=='S'):
    #move position of spliter
    spliter_pos = spliter_pos +1
    error_line = error_line +1
    #push stack to next state
    SLR_stack.append(int(self.SLR_TABLE[current_state][next_input_symbol][1:]))
```

Reduce is occurred when the value of SLR TABLE is like R(19). It means reduce by the rule number 19. The number of right hand side of rule is popped out from our stack. If right hand side is epsilon, popped out nothing. And revise our total input list, and also revise position of splitter because the length of total input list is changed.

Let's move on next step. We need to update current state that is decided by top of the stack. And, we check it is acceptable. If it includes start dummy symbol 'S', then it is accepted. If not, do next step GOTO. Add next state into our stack.

```
#reduce
elif (self.SLR_TABLE[current_state][next_input_symbol][0]=='R'):
    #remove ( , ) to use int
    buf_string = self.SLR_TABLE[current_state][next_input_symbol][1:].replace("(","")
    buf_string = buf_string.replace(")","")
    #get rule , type is list
    buf_rule = self.RULES[buf_string].split()
    buf_length = len(buf_rule) - 2 # ex) 'STMT → VDECL' , we only need VDECL
    #revise terminal list
    for i in range(buf_length):
        if (buf_rule[2] != 'epsilon'):#if not epsilon
            #pop out from stack
            SLR_stack.pop()
            self.terminal_list.pop(spliter_pos - i - 1)
    if (buf_rule[2] != 'epsilon'):#if not epsilon
        spliter_pos = spliter_pos - buf_length +1
    else:#if epsilon
        spliter_pos = spliter_pos+1
    #revise terminal list
    self.terminal_list.insert(spliter_pos-1,buf_rule[0])
    current_state = SLR_stack[-1]
    #Print for debugging
    #print(self.terminal_list)
    if((buf_rule[0] =='S') and (len(self.terminal_list)==2) and (spliter_pos==1)):
        return True, ''
    if buf_rule[0] not in self.SLR_TABLE[current_state].keys():
        report = "Error occurred in line "+str(error_line) + ", " + self.list_for_error_check[error_line-1]
        print(report)
        return False, report
    SLR_stack.append(self.SLR_TABLE[current_state][buf_rule[0]])
```

TEST CASES & RESULT

1. Accept Cases

- ✓ CASE1

Input

```
int main(int arc){

    int num1 = 0;
    int num2 = 0;
    int tmp = 10;
    char welcom = "hello world";

    if( a > b){
        tmp = tmp/2;
        for(i = 0 ; i < 10; i = i + 1){
            i = i*i;
        }
    }
    else{
        while(num2<10){
            num2 = num2 + 1;
        }
    }

    return 0;
}
```

Result

```
junhyuckwoo ~/Documents/CAU/test python3 syntax_analyzer.py code1.out
Accepted
```

✓ CASE2

Input

```
int main(int arc){

    float num1 = 10.0;
    float num2 = 3.0;
    float result;

    result = num1 + num2;

    return result;
}
```

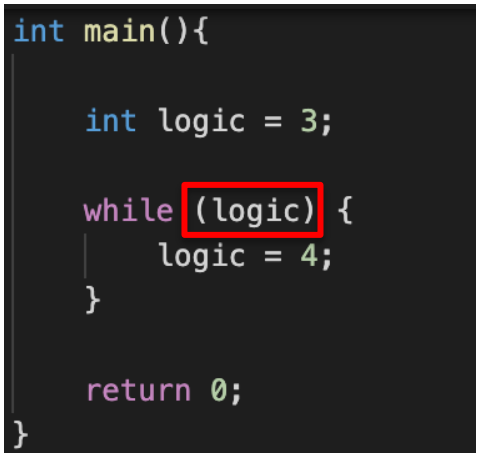
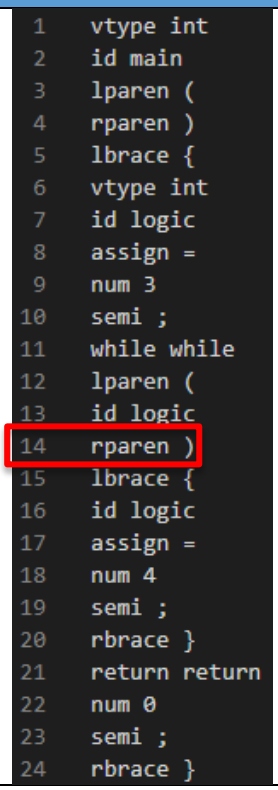
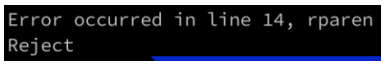
Result

```
junhyuckwoo ~/Documents/CAU/test python3 syntax_analyzer.py code2.out
Accepted
```

2. Reject Cases

We tested various cases. Please check the other cases which are in the TestCase directory.

- ✓ WHILE

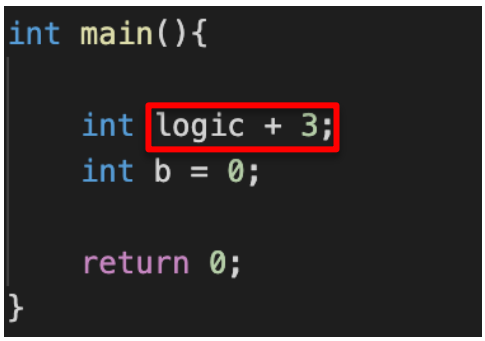
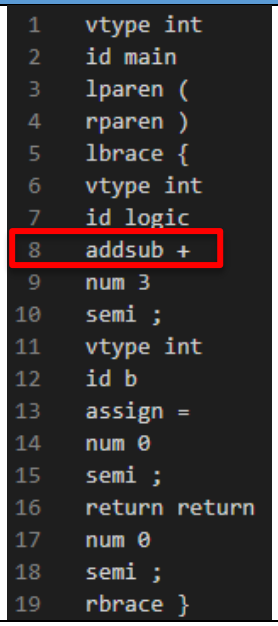
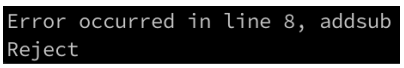
Input	Result_lexical	Result
		

```
int main(){

    int logic = 3;

    while (logic) {
        logic = 4;
    }

    return 0;
}
```

```
1   vtype int
2   id main
3   lparen (
4   rparen )
5   lbrace {
6   vtype int
7   id logic
8   assign =
9   num 3
10  semi ;
11  while while
12  lparen (
13  id logic
14  rparen )
15  lbrace {
16  id logic
17  assign =
18  num 4
19  semi ;
20  rbrace }
21  return return
22  num 0
23  semi ;
24  rbrace }
```

```
Error occurred in line 14, rparen
Reject
```

- ✓ ADDSUB

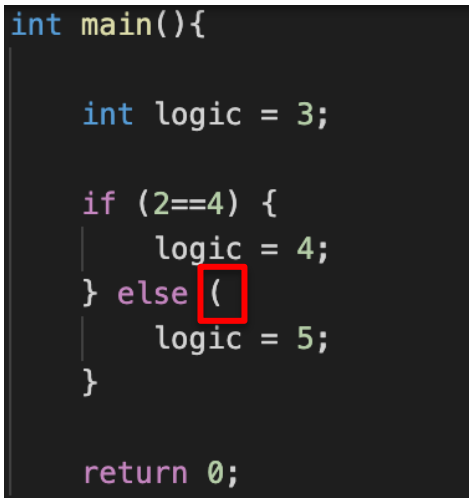
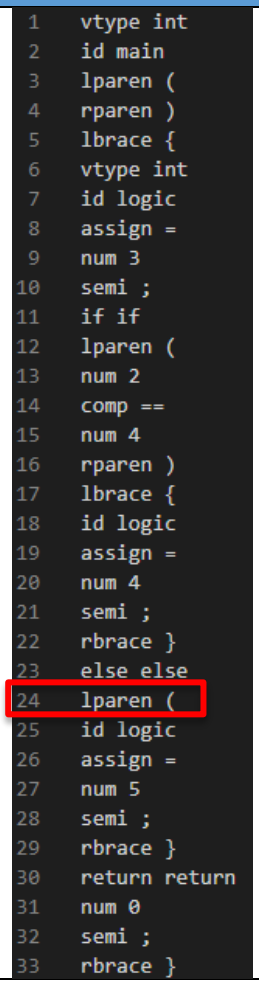
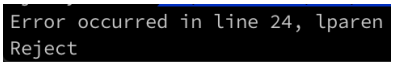
Input	Result_lexical	Result
		

```
int main(){

    int logic + 3;
    int b = 0;

    return 0;
}
```

```
1   vtype int
2   id main
3   lparen (
4   rparen )
5   lbrace {
6   vtype int
7   id logic
8   addsub +
9   num 3
10  semi ;
11  vtype int
12  id b
13  assign =
14  num 0
15  semi ;
16  return return
17  num 0
18  semi ;
19  rbrace }
```

```
Error occurred in line 8, addsub
Reject
```

✓ IF - ELSE

Input	Result_lexical	Result
		

```
int main(){

    int logic = 3;

    if (2==4) {
        logic = 4;
    } else (
        logic = 5;
    }

    return 0;
```

```
1   vtype int
2   id main
3   lparen (
4   rparen )
5   lbrace {
6   vtype int
7   id logic
8   assign =
9   num 3
10  semi ;
11  if if
12  lparen (
13  num 2
14  comp ==
15  num 4
16  rparen )
17  lbrace {
18  id logic
19  assign =
20  num 4
21  semi ;
22  rbrace }
23  else else
24  lparen (
25  id logic
26  assign =
27  num 5
28  semi ;
29  rbrace }
30  return return
31  num 0
32  semi ;
33  rbrace }
```

```
Error occurred in line 24, lparen
Reject
```

APPENDIX

1. First Set

first(RETURN) = first(return FACTOR semi) = {return}

first(FACTOR) = first(lparen EXPR rparen) ∪ first(id) ∪ First(num) ∪ First(float)
= {lparen} ∪ {id} ∪ {num} ∪ {float} = {lparen, id, num, float}

first(TERM) = First(FACTOR multdiv TERM) ∪ first(FACTOR) = first(FACTOR)
= {lparen, id, num, float}

first(COND) = first(FACTOR comp FACTOR) = first(FACTOR) = {lparen, id, num, float}

first(EXPR) = first(TERM addsub EXPR) ∪ first(TERM) = first(TERM) = {lparen, id, num, float}

first(RHS) = first(EXPR) ∪ first(literal) = {lparen, id, num, float} ∪ {literal}
= {lparen, id, num, float, literal}

first(ELSE) = first(else lbrace BLOCK rbrace) ∪ first(ε) = {else} ∪ {ε} = {else, ε}

first(STMT) = first(for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace)
∪ first(while lparen COND rparen lbrace BLOCK rbrace)
∪ first(if lparen COND rparen lbrace BLOCK rbrace ELSE)
∪ first(VDECL) ∪ first(ASSIGN semi)
= first(for) ∪ first(while) ∪ first(if) ∪ first(VDECL) ∪ first(ASSIGN)
= {for, while, if, vtype, id}

first(VDECL) = first(vtype id semi) ∪ first(vtype ASSIGN semi) = first(vtype) = {vtype}

first(ASSIGN) = first(id assign RHS) = first(id) = {id}

first(BLOCK) = first(STMT BLOCK) ∪ first(ε) = first(STMT) ∪ {ε}
= {for, while, if, vtype, id, ε}

first(MOREARGS) = first(comma vtype id MOREARGS) ∪ first(ε) = first(comma) ∪ {ε} = {comma, ε}

first(ARG) = first(vtype id MOREARGS) ∪ first(ε) = first(vtype) ∪ {ε} = {vtype, ε}

first(FDECL) = first(vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace)
= first(vtype) = {vtype}

first(CODE) = first(VDECL CODE) ∪ first(FDECL CODE) ∪ first(ε)
= first(VDECL) ∪ first(FDECL) ∪ {ε}
= {vtype, ε}

first(GOAL) = first(CODE) = {vtype, ε}

2. Follow Set

Follow(GOAL) = {$}

Follow(CODE) = Follow(GOAL) = {$}

Follow(VDECL) = first(CODE) - {ε} ∪ Follow(STMT) ∪ Follow(CODE)
= {vtype} ∪ {lbrace, return, for, while, if, vtype, id} ∪ {$} = {vtype, lbrace, return, for, while, if, id, $}

Follow(ASSIGN) = first(semi) ∪ first(rparen) = {semi rparen}

Follow(FDECL) = first(CODE) - {ε} ∪ Follow(CODE) = {vtype} ∪ {$} = {vtype, $}

Follow(ARG) = first(rparen) = {rparen}

Follow(MOREARGS) = Follow(ARG) = {rparen}

Follow(BLOCK) = first(rbrace) ∪ first(RETURN) - {ε}
= {rbrace} ∪ {return} = {rbrace, return}

Follow(STMT) = first(BLOCK) - {ε} ∪ Follow(BLOCK)
= {id, while, if, vtype, id} ∪ {lbrace, return} = {lbrace, return, for, while, if, vtype, id}

Follow(ELSE) = Follow(STMT)
= {rbrace, return, for, while, if, vtype, id}

Follow(RHS) = Follow(ASSIGN) = {semi, rparen}

Follow(EXPR) = first(rparen) ∪ Follow(RHS)
= {rparen} ∪ {semi, rparen} = {rparen, semi}

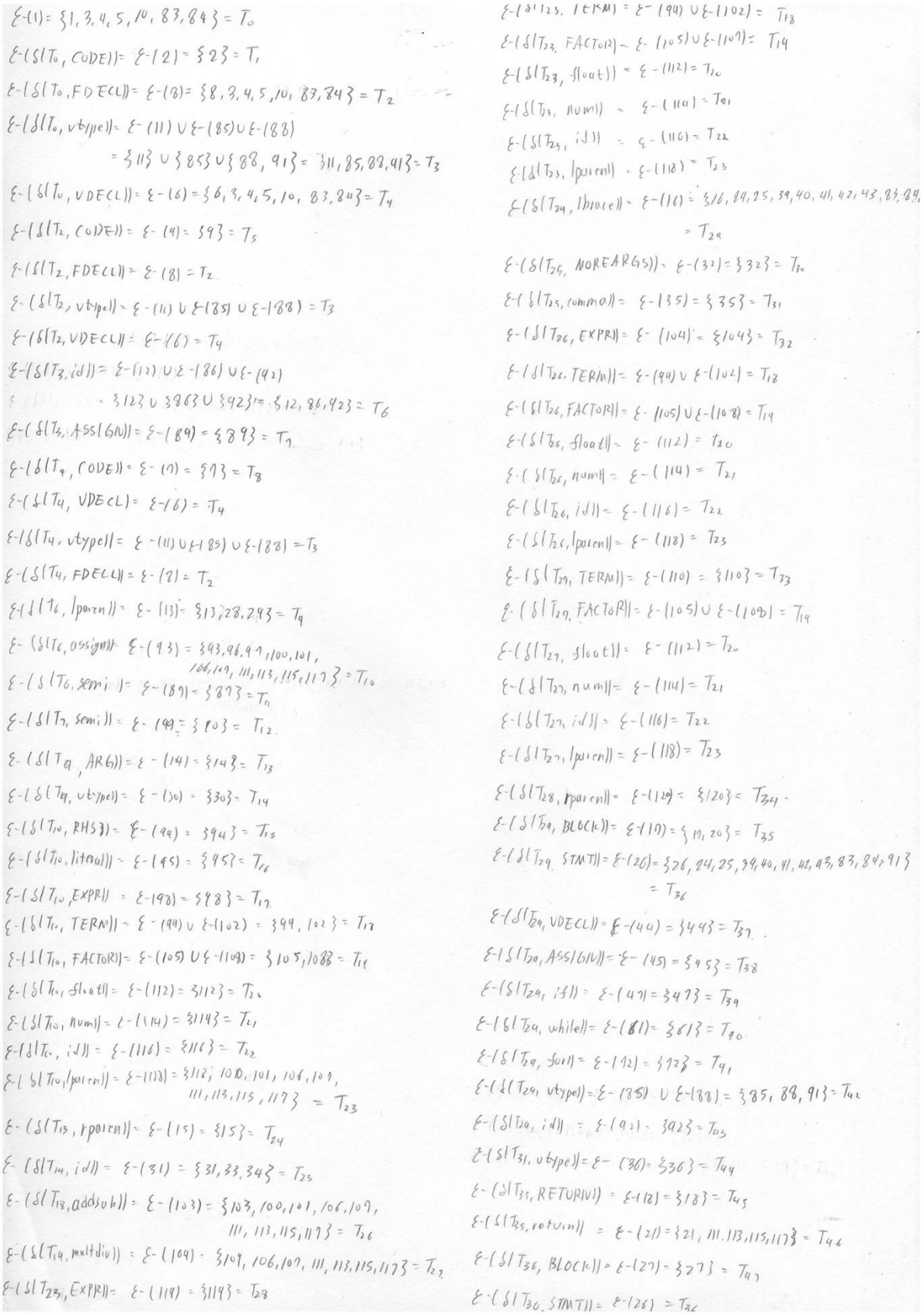
Follow(TERM) = first(addsub) ∪ Follow(EXPR)
= {addsub} ∪ {semi, rparen} = {addsub, semi, rparen}

Follow(FACTOR) = first(comp) ∪ first(multdiv) ∪ first(semi) ∪ Follow(TERM) ∪ Follow(COND)
= {comp} ∪ {multdiv} ∪ {semi} ∪ {addsub, semi rparen} ∪ {semi, rparen}
= {comp, multdiv, addsub, semi, rparen}

Follow(COND) = first(rparen) ∪ first(semi) = {semi, rparen}

Follow(RETURN) = first(rbrace) = {rbrace}

3. NFA to DFA



$\varepsilon(1) = \{1, 3, 4, 5, 10, 83, 84\} = T_0$

$\varepsilon(\delta(T_0, CODE)) = \varepsilon(2) = \{2\} = T_1$

$\varepsilon(\delta(T_0, FDECL)) = \varepsilon(8) = \{8, 3, 4, 5, 10, 83, 84\} = T_2$

$\varepsilon(\delta(T_0, vtype)) = \varepsilon(11) \cup \varepsilon(85) \cup \varepsilon(88)$

$= \{11\} \cup \{85\} \cup \{88, 91\} = \{11, 85, 88, 91\} = T_3$

$\varepsilon(\delta(T_0, VDECL)) = \varepsilon(16) = \{6, 3, 4, 5, 10, 83, 84\} = T_4$

$\varepsilon(\delta(T_1, CODE)) = \varepsilon(9) = \{9\} = T_5$

$\varepsilon(\delta(T_2, FDECL)) = \varepsilon(8) = T_2$

$\varepsilon(\delta(T_2, vtype)) = \varepsilon(11) \cup \varepsilon(85) \cup \varepsilon(88) = T_3$

$\varepsilon(\delta(T_2, VDECL)) = \varepsilon(16) = T_4$

$\varepsilon(\delta(T_3, id)) = \varepsilon(12) \cup \varepsilon(86) \cup \varepsilon(42)$

$= \{12\} \cup \{86\} \cup \{42\} = \{12, 86, 42\} = T_6$

$\varepsilon(\delta(T_3, ASSIGN)) = \varepsilon(89) = \{89\} = T_7$

$\varepsilon(\delta(T_4, CODE)) = \varepsilon(10) = \{9\} = T_8$

$\varepsilon(\delta(T_4, VDECL)) = \varepsilon(16) = T_4$

$\varepsilon(\delta(T_4, vtype)) = \varepsilon(11) \cup \varepsilon(85) \cup \varepsilon(88) = T_3$

$\varepsilon(\delta(T_4, FDECL)) = \varepsilon(8) = T_2$

$\varepsilon(\delta(T_6, lparen)) = \varepsilon(13) = \{13, 28, 29\} = T_9$

$\varepsilon(\delta(T_6, assign)) = \varepsilon(93) = \{93, 96, 97, 100, 101, 106, 109, 111, 113, 115, 117\} = T_{10}$

$\varepsilon(\delta(T_6, semi)) = \varepsilon(89) = \{89\} = T_{11}$

$\varepsilon(\delta(T_7, semi)) = \varepsilon(90) = \{90\} = T_{12}$

$\varepsilon(\delta(T_9, ARG)) = \varepsilon(14) = \{14\} = T_{13}$

$\varepsilon(\delta(T_9, vtype)) = \varepsilon(30) = \{30\} = T_{14}$

$\varepsilon(\delta(T_{10}, RHS)) = \varepsilon(94) = \{94\} = T_{15}$

$\varepsilon(\delta(T_{10}, literal)) = \varepsilon(95) = \{95\} = T_{16}$

$\varepsilon(\delta(T_{10}, EXPR)) = \varepsilon(98) = \{98\} = T_{17}$

$\varepsilon(\delta(T_{10}, TERM)) = \varepsilon(99) \cup \varepsilon(102) = \{99, 102\} = T_{18}$

$\varepsilon(\delta(T_{10}, FACTOR)) = \varepsilon(105) \cup \varepsilon(108) = \{105, 108\} = T_{19}$

$\varepsilon(\delta(T_{10}, float)) = \varepsilon(112) = \{112\} = T_{20}$

$\varepsilon(\delta(T_{10}, num)) = \varepsilon(114) = \{114\} = T_{21}$

$\varepsilon(\delta(T_{10}, id)) = \varepsilon(116) = \{116\} = T_{22}$

$\varepsilon(\delta(T_{10}, lparen)) = \varepsilon(118) = \{118, 100, 101, 106, 109, 111, 113, 115, 117\} = T_{23}$

$\varepsilon(\delta(T_{13}, rparen)) = \varepsilon(15) = \{15\} = T_{24}$

$\varepsilon(\delta(T_{14}, id)) = \varepsilon(31) = \{31, 33, 34\} = T_{25}$

$\varepsilon(\delta(T_{15}, addsub)) = \varepsilon(103) = \{103, 100, 101, 106, 109, 111, 113, 115, 117\} = T_{26}$

$\varepsilon(\delta(T_{16}, multdiv)) = \varepsilon(109) = \{109, 106, 109, 111, 113, 115, 117\} = T_{27}$

$\varepsilon(\delta(T_{23}, EXPR)) = \varepsilon(119) = \{119\} = T_{28}$

$\varepsilon(\delta(T_{23}, TERM)) = \varepsilon(99) \cup \varepsilon(102) = T_{18}$

$\varepsilon(\delta(T_{23}, FACTOR)) = \varepsilon(105) \cup \varepsilon(108) = T_{19}$

$\varepsilon(\delta(T_{23}, float)) = \varepsilon(112) = T_{20}$

$\varepsilon(\delta(T_{23}, num)) = \varepsilon(114) = T_{21}$

$\varepsilon(\delta(T_{23}, id)) = \varepsilon(116) = T_{22}$

$\varepsilon(\delta(T_{23}, lparen)) = \varepsilon(118) = T_{23}$

$\varepsilon(\delta(T_{24}, lbrace)) = \varepsilon(16) = \{16, 84, 25, 39, 40, 41, 42, 43, 83, 84, 91\} = T_{29}$

$\varepsilon(\delta(T_{26}, NOREARGS)) = \varepsilon(32) = \{32\} = T_{30}$

$\varepsilon(\delta(T_{25}, comma)) = \varepsilon(35) = \{35\} = T_{31}$

$\varepsilon(\delta(T_{26}, EXPR)) = \varepsilon(104) = \{104\} = T_{32}$

$\varepsilon(\delta(T_{26}, TERM)) = \varepsilon(99) \cup \varepsilon(102) = T_{18}$

$\varepsilon(\delta(T_{26}, FACTOR)) = \varepsilon(105) \cup \varepsilon(108) = T_{19}$

$\varepsilon(\delta(T_{26}, float)) = \varepsilon(112) = T_{20}$

$\varepsilon(\delta(T_{26}, num)) = \varepsilon(114) = T_{21}$

$\varepsilon(\delta(T_{26}, id)) = \varepsilon(116) = T_{22}$

$\varepsilon(\delta(T_{26}, lparen)) = \varepsilon(118) = T_{23}$

$\varepsilon(\delta(T_{29}, TERM)) = \varepsilon(110) = \{110\} = T_{33}$

$\varepsilon(\delta(T_{29}, FACTOR)) = \varepsilon(105) \cup \varepsilon(108) = T_{19}$

$\varepsilon(\delta(T_{27}, float)) = \varepsilon(112) = T_{20}$

$\varepsilon(\delta(T_{27}, num)) = \varepsilon(114) = T_{21}$

$\varepsilon(\delta(T_{27}, id)) = \varepsilon(116) = T_{22}$

$\varepsilon(\delta(T_{27}, lparen)) = \varepsilon(118) = T_{23}$

$\varepsilon(\delta(T_{28}, rparen)) = \varepsilon(120) = \{120\} = T_{34}$

$\varepsilon(\delta(T_{29}, BLOCK)) = \varepsilon(17) = \{17, 20\} = T_{35}$

$\varepsilon(\delta(T_{29}, STMT)) = \varepsilon(26) = \{26, 24, 25, 39, 40, 41, 42, 43, 83, 84, 91\} = T_{36}$

$\varepsilon(\delta(T_{29}, VDECL)) = \varepsilon(44) = \{44\} = T_{37}$

$\varepsilon(\delta(T_{29}, ASSIGN)) = \varepsilon(45) = \{45\} = T_{38}$

$\varepsilon(\delta(T_{29}, id)) = \varepsilon(47) = \{47\} = T_{39}$

$\varepsilon(\delta(T_{29}, while)) = \varepsilon(81) = \{81\} = T_{40}$

$\varepsilon(\delta(T_{29}, semi)) = \varepsilon(112) = \{112\} = T_{41}$

$\varepsilon(\delta(T_{29}, vtype)) = \varepsilon(85) \cup \varepsilon(88) = \{85, 88, 91\} = T_{42}$

$\varepsilon(\delta(T_{29}, id)) = \varepsilon(92) = \{92\} = T_{43}$

$\varepsilon(\delta(T_{31}, vtype)) = \varepsilon(36) = \{36\} = T_{44}$

$\varepsilon(\delta(T_{35}, RETURN)) = \varepsilon(18) = \{18\} = T_{45}$

$\varepsilon(\delta(T_{35}, return)) = \varepsilon(21) = \{21, 111, 113, 115, 117\} = T_{46}$

$\varepsilon(\delta(T_{36}, BLOCK)) = \varepsilon(27) = \{27\} = T_{47}$

$\varepsilon(\delta(T_{30}, STMT)) = \varepsilon(26) = T_{36}$

$\mathcal{E}(\delta(T_{36}, VDECL)) = \mathcal{E}(44) = T_{37}$

$\mathcal{E}(\delta(T_{36}, ASSIGN)) = \mathcal{E}(45) = T_{38}$

$\mathcal{E}(\delta(T_{36}, if)) = \mathcal{E}(49) = \{49\} = T_{39}$

$\mathcal{E}(\delta(T_{36}, while)) = \mathcal{E}(61) = \{61\} = T_{40}$

$\mathcal{E}(\delta(T_{36}, for)) = \mathcal{E}(92) = \{92\} = T_{41}$

$\mathcal{E}(\delta(T_{36}, vtypeof)) = \mathcal{E}(85) \cup \mathcal{E}(88) = T_{42}$

$\mathcal{E}(\delta(T_{36}, id)) = \mathcal{E}(92) = \{92\} = T_{43}$

$\mathcal{E}(\delta(T_{38}, semi)) = \mathcal{E}(46) = \{46\} = T_{48}$

$\mathcal{E}(\delta(T_{39}, lparen)) = \mathcal{E}(48) = \{48, 68, 111, 113, 115, 117\} = T_{49}$

$\mathcal{E}(\delta(T_{40}, lparen)) = \mathcal{E}(62) = \{62, 68, 111, 113, 115, 117\} = T_{50}$

$\mathcal{E}(\delta(T_{41}, lparen)) = \mathcal{E}(93) = \{93, 94\} = T_{51}$

$\mathcal{E}(\delta(T_{42}, id)) = \mathcal{E}(86) \cup \mathcal{E}(92) = \{86\} \cup \{92\}$

$= \{86, 92\} = T_{52}$

$\mathcal{E}(\delta(T_{42}, ASSIGN)) = \mathcal{E}(89) = T_{7}$

$\mathcal{E}(\delta(T_{43}, assign)) = \mathcal{E}(93) = T_{60}$

$\mathcal{E}(\delta(T_{44}, id)) = \mathcal{E}(37) = \{37, 33, 34\} = T_{53}$

$\mathcal{E}(\delta(T_{45}, lbrace)) = \mathcal{E}(119) = \{19\} = T_{54}$

$\mathcal{E}(\delta(T_{46}, FACTOR)) = \mathcal{E}(22) = \{22\} = T_{55}$

$\mathcal{E}(\delta(T_{46}, float)) = \mathcal{E}(112) = T_{20}$

$\mathcal{E}(\delta(T_{46}, num)) = \mathcal{E}(114) = T_{21}$

$\mathcal{E}(\delta(T_{46}, id)) = \mathcal{E}(116) = T_{22}$

$\mathcal{E}(\delta(T_{46}, lparen)) = \mathcal{E}(118) = T_{23}$

$\mathcal{E}(\delta(T_{49}, COND)) = \mathcal{E}(44) = \{44\} = T_{56}$

$\mathcal{E}(\delta(T_{49}, FACTOR)) = \mathcal{E}(69) = \{69\} = T_{57}$

$\mathcal{E}(\delta(T_{49}, float)) = \mathcal{E}(112) = T_{20}$

$\mathcal{E}(\delta(T_{49}, num)) = \mathcal{E}(114) = T_{21}$

$\mathcal{E}(\delta(T_{49}, id)) = \mathcal{E}(116) = T_{22}$

$\mathcal{E}(\delta(T_{49}, lparen)) = \mathcal{E}(118) = T_{23}$

$\mathcal{E}(\delta(T_{50}, COND)) = \mathcal{E}(63) = \{63\} = T_{58}$

$\mathcal{E}(\delta(T_{50}, FACTOR)) = \mathcal{E}(19) = T_{59}$

$\mathcal{E}(\delta(T_{50}, float)) = \mathcal{E}(112) = T_{20}$

$\mathcal{E}(\delta(T_{50}, num)) = \mathcal{E}(114) = T_{21}$

$\mathcal{E}(\delta(T_{50}, id)) = \mathcal{E}(116) = T_{22}$

$\mathcal{E}(\delta(T_{50}, lparen)) = \mathcal{E}(118) = T_{23}$

$\mathcal{E}(\delta(T_{51}, ASSIGN)) = \mathcal{E}(94) = \{94\} = T_{54}$

$\mathcal{E}(\delta(T_{51}, semi)) = \mathcal{E}(92) = T_{43}$

$\mathcal{E}(\delta(T_{52}, semi)) = \mathcal{E}(89) = T_{11}$

$\mathcal{E}(\delta(T_{52}, assign)) = \mathcal{E}(93) = T_{60}$

$\mathcal{E}(\delta(T_{53}, MOREARGS)) = \mathcal{E}(38) = \{38\} = T_{60}$

$\mathcal{E}(\delta(T_{53}, comma)) = \mathcal{E}(35) = T_{31}$

$\mathcal{E}(\delta(T_{55}, semi)) = \mathcal{E}(23) = \{23\} = T_{61}$

$\mathcal{E}(\delta(T_{56}, rparen)) = \mathcal{E}(50) = \{50\} = T_{62}$

$\mathcal{E}(\delta(T_{57}, comp)) = \mathcal{E}(90) = \{78, 111, 113, 115, 117\} = T_{63}$

$\mathcal{E}(\delta(T_{58}, rparen)) = \mathcal{E}(64) = \{64\} = T_{64}$

$\mathcal{E}(\delta(T_{59}, semi)) = \mathcal{E}(95) = \{95, 68, 111, 113, 115, 117\} = T_{65}$

$\mathcal{E}(\delta(T_{62}, lbrace)) = \mathcal{E}(51) = \{51, 24, 25, 39, 40, 41, 42, 43, 83, 84, 9$

$= T_{66}$

$\mathcal{E}(\delta(T_{65}, FACTOR)) = \mathcal{E}(91) = \{91\} = T_{67}$

$\mathcal{E}(\delta(T_{63}, float)) = \mathcal{E}(112) = T_{20}$

$\mathcal{E}(\delta(T_{63}, num)) = \mathcal{E}(114) = T_{21}$

$\mathcal{E}(\delta(T_{63}, id)) = \mathcal{E}(116) = T_{22}$

$\mathcal{E}(\delta(T_{64}, lparen)) = \mathcal{E}(118) = T_{23}$

$\mathcal{E}(\delta(T_{66}, lbrace)) = \mathcal{E}(65) = \{65, 24, 25, 39, 40, 41, 42, 43, 83, 84, 9$

$= T_{68}$

$\mathcal{E}(\delta(T_{65}, COND)) = \mathcal{E}(96) = \{96\} = T_{69}$

$\mathcal{E}(\delta(T_{65}, FACTOR)) = \mathcal{E}(69) = T_{59}$

$\mathcal{E}(\delta(T_{65}, float)) = \mathcal{E}(112) = T_{20}$

$\mathcal{E}(\delta(T_{65}, num)) = \mathcal{E}(114) = T_{21}$

$\mathcal{E}(\delta(T_{65}, id)) = \mathcal{E}(116) = T_{22}$

$\mathcal{E}(\delta(T_{65}, lparen)) = \mathcal{E}(117) = T_{23}$

$\mathcal{E}(\delta(T_{66}, BLOCK)) = \mathcal{E}(52) = \{52\} = T_{70}$

$\mathcal{E}(\delta(T_{66}, STMT)) = \mathcal{E}(26) = T_{36}$

$\mathcal{E}(\delta(T_{66}, VDECL)) = \mathcal{E}(44) = T_{37}$

$\mathcal{E}(\delta(T_{66}, ASSIGN)) = \mathcal{E}(45) = T_{38}$

$\mathcal{E}(\delta(T_{66}, if)) = \mathcal{E}(49) = T_{39}$

$\mathcal{E}(\delta(T_{66}, while)) = \mathcal{E}(61) = T_{40}$

$\mathcal{E}(\delta(T_{66}, for)) = \mathcal{E}(92) = T_{41}$

$\mathcal{E}(\delta(T_{66}, vtypeof)) = \mathcal{E}(85) \cup \mathcal{E}(88) = T_{42}$

$\mathcal{E}(\delta(T_{66}, id)) = \mathcal{E}(92) = T_{43}$

$\mathcal{E}(\delta(T_{68}, BLOCK)) = \mathcal{E}(66) = \{66\} = T_{71}$

$\mathcal{E}(\delta(T_{68}, STMT)) = \mathcal{E}(26) = T_{36}$

$\mathcal{E}(\delta(T_{68}, VDECL)) = \mathcal{E}(44) = T_{37}$

$\mathcal{E}(\delta(T_{68}, ASSIGN)) = \mathcal{E}(45) = T_{38}$

$\mathcal{E}(\delta(T_{68}, if)) = \mathcal{E}(47) = T_{39}$

$\mathcal{E}(\delta(T_{68}, while)) = \mathcal{E}(61) = T_{40}$

$\mathcal{E}(\delta(T_{68}, for)) = \mathcal{E}(92) = T_{41}$

$\mathcal{E}(\delta(T_{68}, vtypeof)) = \mathcal{E}(85) \cup \mathcal{E}(88) = T_{42}$

$\mathcal{E}(\delta(T_{68}, id)) = \mathcal{E}(92) = T_{43}$

ξ-(δ(T69, semi)) = ξ-(17) = {77, 91} = T_{72}

ξ-(δ(T70, rbrace)) = ξ-(53) = {53, 55, 58} = T_{73}

ξ-(δ(T71, rbrace)) = ξ-(69) = {67} = T_{74}

ξ-(δ(T72, ASSIGN)) = ξ-(18) = {18} = T_{75} ξ-(δ(T72, id)) = ξ-(42) = T_{43}

ξ-(δ(T73, ELSE)) = ξ-(54) = {54} = T_{76}

ξ-(δ(T73, else)) = ξ-(57) = {57} = T_{77}

ξ-(δ(T75, rparen)) = ξ-(19) = {19} = T_{78}

ξ-(δ(T77, lbrace)) = ξ-(58) = {58, 24, 25, 39, 40, 41, 42, 43, 83, 84, 91} = T_{79}

ξ-(δ(T78, lbrace)) = ξ-(80) = {80, 24, 25, 39, 40, 41, 42, 43, 83, 84, 91} = T_{80}

ξ-(δ(T79, BLOCK)) = ξ-(59) = {59} = T_{81}

ξ-(δ(T79, STMT)) = ξ-(26) = T_{36}

ξ-(δ(T79, VDECL)) = ξ-(44) = T_{37}

ξ-(δ(T79, ASSIGN)) = ξ-(45) = T_{38}

ξ-(δ(T79, id)) = ξ-(47) = T_{39}

ξ-(δ(T79, while)) = ξ-(61) = T_{40}

ξ-(δ(T79, for)) = ξ-(72) = T_{41}

ξ-(δ(T79, vtype)) = ξ-(85) ∪ ξ-(88) = T_{42}

ξ-(δ(T79, id)) = ξ-(42) = T_{43}

ξ-(δ(T81, rbrace)) = ξ-(60) = {60} = T_{83}

ξ-(δ(T82, rbrace)) = ξ-(82) = {82} = T_{84}

4. DFA Node Name List

T0	1 S' -> .CODE 3 CODE ->. 4 CODE -> .FDECL CODE 5 CODE -> .VDECL CODE 10 FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi
T1	2 S' -> CODE.
T2	3 CODE ->. 4 CODE -> .FDECL CODE 5 CODE -> .VDECL CODE 8 CODE -> FDECL. CODE 10 FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi
T3	11 FDECL -> vtype. id lparen ARG rparen lbrace BLOCK RETURN rbrace 85 VDECL -> vtype. id semi 88 VDECL -> vtype. ASSIGN semi 91 ASSIGN -> .id assign RHS
T4	3 CODE ->. 4 CODE -> .FDECL CODE 5 CODE -> .VDECL CODE 6 CODE -> VDECL. CODE 10 FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi
T5	9 CODE -> FDECL CODE.
T6	12 FDECL -> vtype id. lparen ARG rparen lbrace BLOCK RETURN rbrace 86 VDECL -> vtype id. semi 92 ASSIGN -> id. assign RHS
T7	89 VDECL -> vtype ASSIGN. semi
T8	7 CODE -> VDECL CODE.
T9	13 FDECL -> vtype id lparen. ARG rparen lbrace BLOCK RETURN rbrace 28 ARG -> . 29 ARG -> .vtype id MOREARGS
T10	93 ASSIGN -> id assign. RHS 96 RHS -> .literal 97 RHS -> .EXPR 100 EXPR -> .TERM 101 EXPR -> .TERM addsub EXPR 106 TERM -> .FACTOR 107 TERM -> .FACTOR multdiv TERM 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen
T11	87 VDECL -> vtype id semi.
T12	90 VDECL -> vtype ASSIGN semi.
T13	14 FDECL -> vtype id lparen ARG. rparen lbrace BLOCK RETURN rbrace
T14	30 ARG -> vtype. id MOREARGS
T15	94 ASSIGN -> id assign RHS.
T16	95 RHS -> literal.
T17	98 RHS -> EXPR .
T18	99 EXPR -> TERM. 102 EXPR -> TERM. addsub EXPR
T19	105 TERM -> FACTOR. 108 TERM -> FACTOR. multdiv TERM
T20	112 FACTOR -> float.
T21	114 FACTOR -> num.
T22	116 FACTOR -> id.
T23	100 EXPR -> .TERM 101 EXPR -> .TERM addsub EXPR 106 TERM -> .FACTOR 107 TERM -> .FACTOR multdiv TERM 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen 118 FACTOR -> lparen. EXPR rparen
T24	15 FDECL -> vtype id lparen ARG rparen. lbrace BLOCK RETURN rbrace
T25	31 ARG -> vtype id. MOREARGS 33 MOREARGS -> . 34 MOREARGS -> .comma vtype id MOREARGS
T26	100 EXPR -> .TERM 101 EXPR -> .TERM addsub EXPR 103 EXPR -> TERM addsub. EXPR 106 TERM -> .FACTOR 107 TERM -> .FACTOR multdiv TERM 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id

	117 FACTOR -> .lparen EXPR rparen
T27	106 TERM -> .FACTOR 107 TERM -> .FACTOR multdiv TERM 109 TERM -> FACTOR multdiv. TERM 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen
T28	119 FACTOR -> lparen EXPR. rparen
T29	16 FDECL -> vtype id lparen ARG rparen lbrace. BLOCK RETURN rbrace 24 BLOCK -> . 25 BLOCK -> .STMT BLOCK 39 STMT -> .VDECL 40 STMT -> .ASSIGN semi 41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE 42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace 43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi 91 ASSIGN -> .id assign RHS
T30	32 ARG -> vtype id MOREARGS.
T31	35 MOREARGS -> comma. vtype id MOREARGS
T32	104 EXPR -> TERM addsub EXPR.
T33	110 TERM -> FACTOR multdiv TERM.
T34	120 FACTOR -> lparen EXPR rparen.
T35	17 FDECL -> vtype id lparen ARG rparen lbrace BLOCK. RETURN rbrace 20 RETURN -> .return FACTOR semi
T36	24 BLOCK -> . 25 BLOCK -> .STMT BLOCK 26 BLOCK -> STMT. BLOCK 39 STMT -> .VDECL 40 STMT -> .ASSIGN semi 41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE 42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace 43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi 91 ASSIGN -> .id assign RHS
T37	44 STMT -> VDECL.
T38	45 STMT -> ASSIGN. Semi
T39	47 STMT -> if. lparen COND rparen lbrace BLOCK rbrace ELSE
T40	61 STMT -> while. lparen COND rparen lbrace BLOCK rbrace
T41	72 STMT -> for. lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace
T42	85 VDECL -> vtype. id semi 88 VDECL -> vtype. ASSIGN semi 91 ASSIGN -> .id assign RHS
T43	92 ASSIGN -> id. assign RHS
T44	36 MOREARGS -> comma vtype. id MOREARGS
T45	18 FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN. rbrace
T46	21 RETURN -> return. FACTOR semi 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen
T47	27 BLOCK -> STMT BLOCK .
T48	46 STMT -> ASSIGN semi.
T49	48 STMT -> if lparen. COND rparen lbrace BLOCK rbrace ELSE 68 COND -> .FACTOR comp FACTOR 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen
T50	62 STMT -> while lparen. COND rparen lbrace BLOCK rbrace 68 COND -> .FACTOR comp FACTOR 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen
T51	73 STMT -> for lparen. ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 91 ASSIGN -> .id assign RHS
T52	86 VDECL -> vtype id. semi 92 ASSIGN -> id. assign RHS
T53	33 MOREARGS -> . 34 MOREARGS -> .comma vtype id MOREARGS 37 MOREARGS -> comma vtype id. MOREARGS
T54	19 FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace.
T55	22 RETURN -> return FACTOR. semi
T56	49 STMT -> if lparen COND. rparen lbrace BLOCK rbrace ELSE
T57	69 COND -> FACTOR. comp FACTOR
T58	63 STMT -> while lparen COND. rparen lbrace BLOCK rbrace
T59	74 STMT -> for lparen ASSIGN. semi COND semi ASSIGN rparen lbrace BLOCK rbrace
T60	38 MOREARGS -> comma vtype id MOREARGS.

T61	23 RETURN -> return FACTOR semi.
T62	50 STMT -> if lparen COND rparen. lbrace BLOCK rbrace ELSE
T63	70 COND -> FACTOR comp. FACTOR 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen
T64	64 STMT -> while lparen COND rparen. lbrace BLOCK rbrace
T65	68 COND -> .FACTOR comp FACTOR 75 STMT -> for lparen ASSIGN semi. COND semi ASSIGN rparen lbrace BLOCK rbrace 111 FACTOR -> .float 113 FACTOR -> .num 115 FACTOR -> .id 117 FACTOR -> .lparen EXPR rparen
T66	24 BLOCK -> . 25 BLOCK -> .STMT BLOCK 39 STMT -> .VDECL 40 STMT -> .ASSIGN semi 41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE 42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace 43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi 91 ASSIGN -> .id assign RHS
T67	71 COND -> FACTOR comp FACTOR.
T68	24 BLOCK -> . 25 BLOCK -> .STMT BLOCK 39 STMT -> .VDECL 40 STMT -> .ASSIGN semi 41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE 42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace 43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi 91 ASSIGN -> .id assign RHS
T69	76 STMT -> for lparen ASSIGN semi COND. semi ASSIGN rparen lbrace BLOCK rbrace
T70	52 STMT -> if lparen COND rparen lbrace BLOCK. rbrace ELSE
T71	66 STMT -> while lparen COND rparen lbrace BLOCK. Rbrace
T72	77 STMT -> for lparen ASSIGN semi COND semi. ASSIGN rparen lbrace BLOCK rbrace 91 ASSIGN -> .id assign RHS
T73	53 STMT -> if lparen COND rparen lbrace BLOCK rbrace. ELSE 55 ELSE -> . 56 ELSE -> .else lbrace BLOCK rbrace
T74	67 STMT -> while lparen COND rparen lbrace BLOCK rbrace.
T75	78 STMT -> for lparen ASSIGN semi COND semi ASSIGN. rparen lbrace BLOCK rbrace
T76	54 STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE.
T77	57 ELSE -> else. lbrace BLOCK rbrace
T78	79 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen. lbrace BLOCK rbrace
T79	24 BLOCK -> . 25 BLOCK -> .STMT BLOCK 39 STMT -> .VDECL 40 STMT -> .ASSIGN semi 41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE 42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace 43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 58 ELSE -> else lbrace. BLOCK rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi 91 ASSIGN -> .id assign RHS
T80	24 BLOCK -> . 25 BLOCK -> .STMT BLOCK 39 STMT -> .VDECL 40 STMT -> .ASSIGN semi 41 STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE 42 STMT -> .while lparen COND rparen lbrace BLOCK rbrace 43 STMT -> .for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 80 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen lbrace. BLOCK rbrace 83 VDECL -> .vtype ASSIGN semi 84 VDECL -> .vtype id semi 91 ASSIGN -> .id assign RHS
T81	59 ELSE -> else lbrace BLOCK. rbrace
T82	81 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK. rbrace
T83	60 ELSE -> else lbrace BLOCK rbrace.
T84	82 STMT -> for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace.

5. Rules

1. S’ → CODE
2. CODE → VDECL CODE
3. CODE → FDECL CODE
4. CODE → ε
5. VDECL → vtype id semi
6. VDECL → vtype ASSIGN semi
7. ASSIGN → id assign RHS
8. FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
9. ARG → vtype id MOREARGS
10. ARG → ε
11. MOREARGS → comma vtype id MOREARGS
12. MOREARGS → ε
13. BLOCK → STMT BLOCK
14. BLOCK → ε
15. STMT → VDECL
16. STMT → ASSIGN semi
17. STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE
18. STMT → while lparen COND rparen lbrace BLOCK rbrace
19. STMT → for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace
20. ELSE → else lbrace BLOCK rbrace
21. ELSE → ε
22. RHS → EXPR
23. RHS → literal
24. EXPR → TERM addsub EXPR
25. EXPR → TERM
26. TERM → FACTOR multdiv TERM
27. TERM → FACTOR
28. FACTOR → lparen EXPR rparen
29. FACTOR → id
30. FACTOR → num
31. FACTOR → float
32. COND → FACTOR comp FACTOR
33. RETURN → return FACTOR semi