# CS 419 Compiler Projects Form

1. **Instructions to be Followed (for the Hard Copy):**
   1. Fill page **NO 4** with the required Fields:
      1. Project Idea: The **Idea** will be assigned to you .
      2. Team Members NO#: Number of team members **7**
      3. Table:
         * ID: Your FCIH ID.
         * Name: Your **Full Name** as registered on College’s Database.
         * Level & Department: Your **(Current)** level and department.
         * Section(Day-from-to): Your Section Day and time slot.
         * Role: Your role in project (**Team leader** OR **Member**).
         * Fill Page **NO 5 & 6** with your (**Regular Expressions**, **Finite automata**, **Parse trees** and **abstract syntax tree**) respectively.
2. **Minus Policies:**
   1. **Project Policy:** affects all projects members including team leader.
   2. **Member Policy:** affects a member of project’s members.
3. **General Notes:**
   1. Total grade of Project is 15
   2. **Deadline** to register yourself and your team on online form **Tuesday 05/04/2022 at 11:59 PM** after that **-2 Project Policy** will be applied**.**
   3. Once you Registered, **NO modifications** will be done.
   4. Allowed only on registration for team in form, duplication will got **-2 Project Policy.**
   5. Each group will be assigned **an Idea, ID and time slot** for **discussion**.
   6. Each team member and team leader in a team **must work in project’s coding phase** (including implementation of **finite automata and parse trees**).
4. **Discussion Notes:**
   1. **Copied Code** will be got ZERO Without Discussion.
   2. By references to Section 3 (General notes) Point V , **-5 Member policy** will be applied to each team member (including team leader) who does not participate in project coding phase **as well as the team leader who does not report this case**.
   3. Each team member must have **a complete knowledge** about the whole project
   4. **Evaluation** will be **Individual Evaluation** not project Evaluation.
   5. **-2 Project Policy** will be applied in case of being late for assigned discussion time slot
   6. **NO discussion will be repeated under any circumstances.**
   7. At Discussion day, in case of offline discussions, each team must have **Hard Copy form** including (**Finite automata and parse trees of team’s project**).
   8. Discussion Day will be **sent later** .

## Notes about Implementation:

* 1. **.Net or PHP** are only allowed.
  2. The project must be a Web (use latest technologies).
  3. Your code must be uploaded to github before discussion.
  4. **- 5 Project policy** will be applied in case of using **Built-in Method** within implementation of the scanner or parser, you must create your **own methods to match** for ex (your regular Expressions).
  5. Each Project must contain a full functional editor (comment, uncomment, put red line under wrong words, auto complete, navigation to function or class, line NO).
  6. Each Project must contain **two buttons** , one button called “**Scan**” to run scanner and other called “**Parse**” to run parser –parser must take output of scanner to do it’s task.
  7. Each project must contain a button named “**Browse**” that allows us to choose a **file from a disk** that allows us to parse or scan this file **Without Showing what is inside the file** and shows the output.
  8. – 3 Project policy will be applied if the content of the file that is mentioned in point V is opened or viewed.

## Notes about Discussion Testing:

* + There will be two types of Testing :

1. **White Box Testing:** This will be from **Editor**.
2. **Black Box Testing:** This will be from “**Browse**” Button

**Thanks,**

# CS 419 Compiler Project Form

**Project Idea:**

……………………………………………………………….

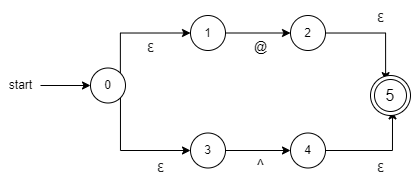
**Team Members NO#:** …………………………………..

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Name** | **Level& Department** | **Section** | **Role** | **Grade** |
| 201900628 | محمد احمد مختار محمد | Level:3  CS | Wed  4:00 pm | Leader |  |
| 201900261 | حسام الدين علي سيد علي | Level:3  CS | Wed  12:00 pm | Member |  |
| 201900518 | عمر عبدربه عبد الحليم عبدلله | Level:3  CS | Wed  2:00 pm | Member |  |
| 201900071 | احمد عمرو إبراهيم عبد السلام | Level:3  CS | Wed  10:00 am | Member |  |
| 201900718 | محمد لبيب مرسي لبيب | Level:3  CS | Wed  4:00 pm | Member |  |
| 201900623 | محمد احمد سيد عبد الرحيم | Level:3  CS | Wed  4:00 pm | Member |  |
| 20180727 | محمد خير عماد محمد | Level:4  CS | Thu  8:00 am | Member |  |

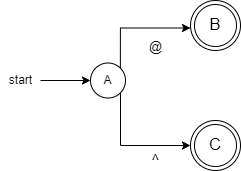
**Regex:**

Start-> (@|^)

**NFA:**



**DFA:**



**Regex:**

Class -> (Type)

**NFA:**



**DFA:**



**Regex:**

Inheritance -> (Infer)

**NFA:**



**DFA:**



**Regex:**

If -> (If)

**NFA:**



**DFA:**

****

**Regex:**

Else-> (Else)

**NFA:**



**DFA:**



**Regex:**

Integer -> (Ipok)

**NFA:**



**DFA:**

****

**Regex:**

SInteger-> (Sipok)

**NFA:**



**DFA:**



**Regex:**

Character-> (Craf)

**NFA:**



**DFA:**

e****

**Regex:**

String-> (Sequence)

**NFA:**



**DFA:**



**Regex:**

Float -> (Ipokf)

**NFA:**



**DFA:**

****

**Regex:**

Sfloat -> (Sipokf)

**NFA:**



**DFA:**



**Regex:**

Void -> (Valueless)

**NFA:**

****

**DFA:**

****

**Regex:**

Boolean -> (Rational)

**NFA:**



**DFA:**



**Regex:**

Break-> (Endthis)

**NFA:**



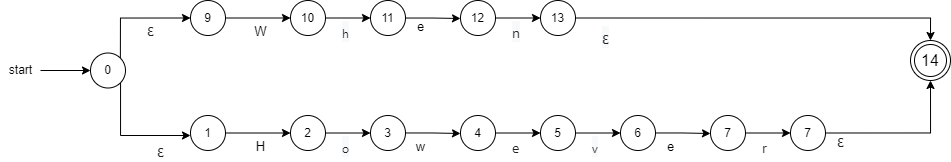
**DFA:**

****

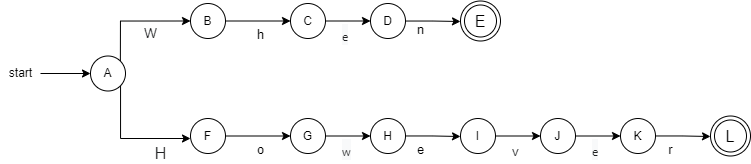
**Regex:**

Loop -> (However|When)

**NFA:**



**DFA:**



**Regex:**

Return -> (Respondwith)

**NFA:**



**DFA:**



**Regex:**

Struct -> (Srap)

**NFA:**



**DFA:**



**Regex:**

Switch -> (Scan)

**NFA:**



**DFA:**



**Regex:**

Case -> (Conditionof)

**NFA:**



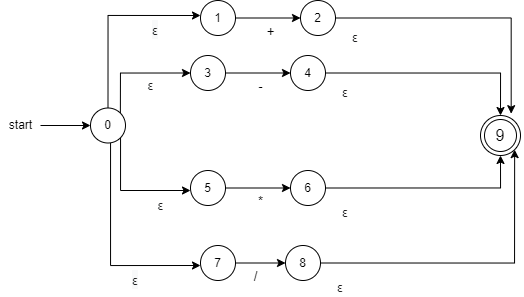
**DFA:**



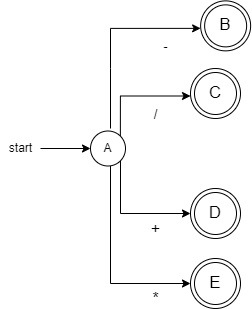
**Regex:**

ArithmeticOP -> (+|-|\*|/)

**NFA:**



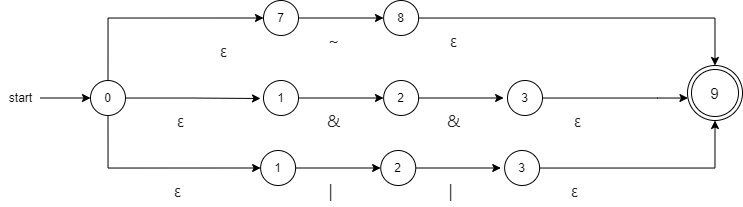
**DFA:**

****

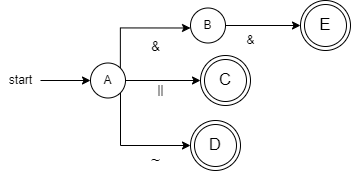
**Regex:**

LogicOP -> (&&| || | ~)

**NFA:**



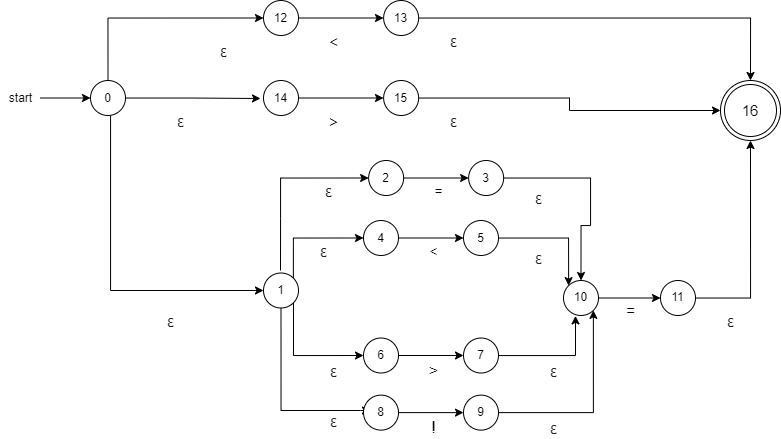
**DFA:**



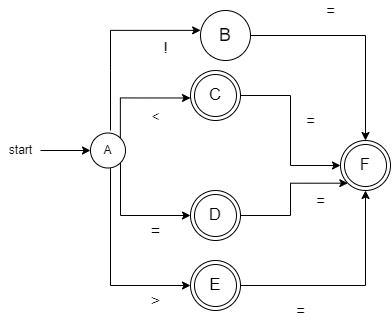
**Regex:**

RelationalOP -> ( = | < | > | ! ) = | < | >

**NFA:**



**DFA:**



**Regex:**

Assignment -> (=)

**NFA:**



**DFA:**



**Regex:**

Access -> (->)

**NFA:**



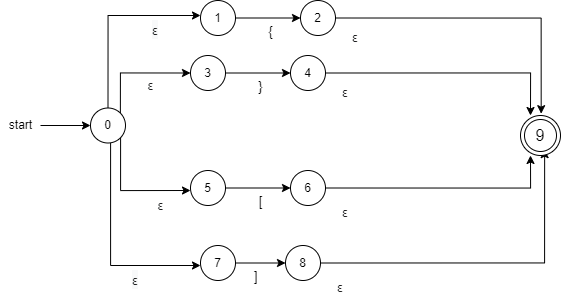
**DFA:**



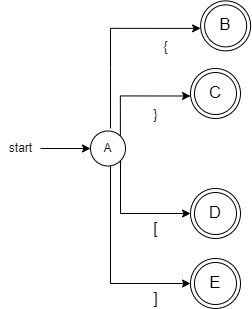
**Regex:**

Braces -> ({|}|[|])

**NFA:**



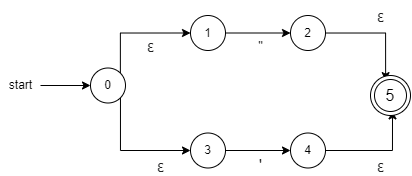
**DFA:**



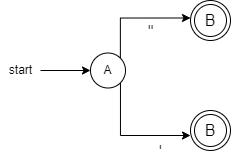
**Regex:**

Quotation -> (“|’)

**NFA:**



**DFA:**



**Regex:**

Inclusion -> (Require)

**NFA:**



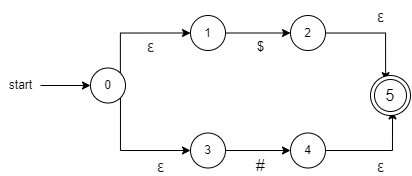
**DFA:**

****

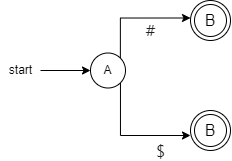
**Regex:**

End -> ($|#)

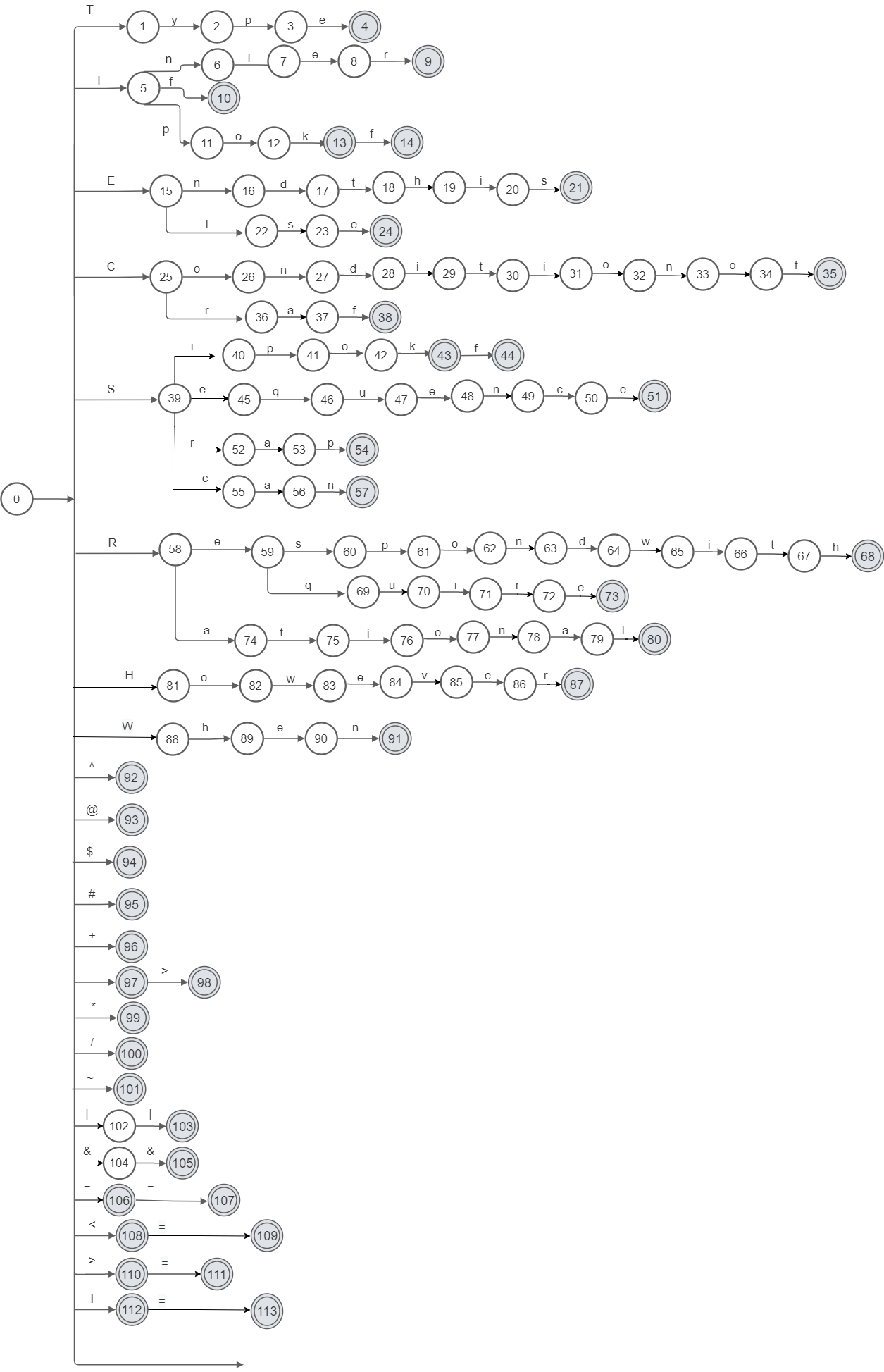
**NFA:**



**DFA:**



Composite DFA



**Parser**

**First step: removing left recursion**

**before**

*10. Non-Empty List→ Type ID | Non-Empty List , Type ID*

*12. ID\_List →ID | ID\_List , ID*

*18.NonEmpty\_Argument\_List →Expression | NonEmpty\_Argument\_List , Expression*

*29. Expression → Term |Expression Add\_Op Term*

*31. Term→Factor| Term Mul\_Op Factor*

**After**

*(10).*

Non-Empty List *→ Type ID Non-Empty List’*

*(Non-Empty List)’ → ,Type ID Non-Empty List’ | €*

*(12).*

*ID\_List→ID ID\_List’*

*ID\_List’→ID ID\_List’|€*

*(18).*

*NonEmpty\_Argument\_List→Expression NonEmpty\_Argument\_List’*

*NonEmpty\_Argument\_List’→ Expression NonEmpty\_Argument\_List’|€*

*(29).*

*Expression → Term Expression’*

*Expression’→ Add\_Op Term Expression’ | €*

*(31).*

*Term → Factor Term’*

*Term’ → Mul\_Op Factor Term’ | €*

**Second step: calculate first**

1. First(Program) -> {@,^}
2. First(Start-Symbols) -> {@,^}
3. First(End-Symbols) -> {$,#}
4. First(ClassDeclaration) -> {Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational}
5. First(Class\_Implementation) -> {Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational,</,\*\*\*,Require,ID,em}
6. First(Method\_Decl) -> {Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational}
7. First(Func\_Decl) -> {Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational}
8. First(Type) -> {Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational}
9. First(ParameterList) -> {em,None,Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational}

10-First(Non-Empty-List)-> {Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational}

10’- First(Non-Empty-List’)->{,,em}

11 - First(Variable\_Decl) ->{em,Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational}

12 - First(ID\_List)={ID}

12’- First(ID\_List’)={, , em}

13- First(Statements)={em,Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational, If \_Statement , However \_Statement , when\_Statement , Respondwith \_ Statement , Endthis}

14-First(Statement)={em,Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational, If \_Statement , However \_Statement , when\_Statement , Respondwith \_ Statement , Endthis}

15-First(Assignment)={em, Ipok,Sipok,Craf,Sequance,Ipokf,Sipokf,Valueless,Rational}

16 – First(Func \_Call ) = {ID}

17 – First(Argument\_List) = {em, ID,Number}

18 – First(NonEmpty\_Argument\_List)={ID,Number}

18’- First(NonEmpty\_Argument\_List’)={em,ID,Number}

19-First(Block Statements)={em}

20-First(If \_Statement)={if}

21-First(Condition \_Expression) = {ID,Number}

22- First(Condition \_Op) = {&& , ||}

23- First(Condition)={ID,Number}

24- First(Comparison \_Op) = {== , != , > , >= , < , <=}

25- First(However \_Statement) = {However}

26- First(when \_Statement) = {when }

27- First(Respondwith \_Statement)={ Respondwith , return}

28- First(Endthis \_Statement) = {Endthis}

29- First(Expression) = {ID, Number}

29’-First(Expression’) = {+,-,em}

30-First(Add\_Op) = {+, -}

31-First(Term) ={ID, Number}

31’-First(Term’) ={em,\*, /}

32-First(Mul\_Op) = {\* , /}

33-First(Factor) = {ID, Number}

34-First(Comment) = {</ , \*\*\*}

35-First(Require\_command) = {Require}

36-First(F\_name) = {STR}

**Third step: calculate follow**

1-follow(Program)={$}

2- follow(Start-Symbols)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational }

3- follow(End-Symbols)={$}

4- follow(ClassDeclaration)={$,#}

5- follow(Class\_Implementation)={ } }

6- follow(Method\_Decl)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational ,</,\*\*\*,Require,ID}

7- follow(Func Decl)={ ; , { }

8- follow(Type)={ID}

9- follow(ParameterList)={)}

10-follow(Non\_Empty List)={)}

10’- follow(Non\_Empty List’)={)}

11-follow(Variable\_Decl)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,</,\*\*\*,Require,ID,},if,However,When,Respondwith,return,Endthis,=}

12-follow(ID\_List)={ ; , [ }

12’- follow(ID\_List’)={ ; , [ }

13-follow(Statements)={ } }

14-follow(Statement)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,if,However,when,Respondwith,return,Endthis,}}

15-follow(Assignment)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,if,However,when,Respondwith,return,Endthis,}}

16-follow(Func\_Call)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,</,\*\*\*,Require,ID}

17-follow(Argument)={)}

18-follow(NonEmpty\_Argument\_List)={)}

18’-follow(NonEmpty\_Argument\_List’)={)}

19-follow(Block Statements)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,if,However,when,Respondwith,return,Endthis,},else}

20-follow(If \_Statement)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless Rational,if,However,when,Respondwith,return,Endthis,}}

21-follow(Condition \_Expression)={)}

22-follow(Condition \_Op)={ID,Number}

23-follow(Condition)={),&&,||}

24-follow(Comparison \_Op)={ID,Number}

25-follow(However \_Statement)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,if,However,when,Respondwith,return,Endthis,}}

26-follow(when \_Statement)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,if,However,when,Respondwith,return,Endthis,}}

27-follow(Respondwith \_Statement)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,if,However,when,Respondwith,return,Endthis,}}

28-follow(Endthis \_Statement)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,if,However,when,Respondwith,return,Endthis,}}

29-follow(Expression)={),;, == , != , > , >= , < , <= ,ID,Number}

29’- follow(Expression’)={),;, == , != , > , >= , < , <= ,ID,Number}

30- follow(Add\_Op) ={ID,Number}

31- follow(Term)={),;, == , != , > , >= , < , <= ,ID,Number}

31’- follow(Term’)={),;, == , != , > , >= , < , <= ,ID,Number}

32- follow(Mul\_Op)={ID,Number}

33- follow(Factor)={),;, == , != , > , >= , < , <= ,ID,Number,\*,/}

34- follow(Comment)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,</,\*\*\*,Require,ID}

35- follow(Require\_command)={ Ipok ,Sipok ,Craf ,Sequence ,Ipokf ,Sipokf ,Valueless ,Rational,</,\*\*\*,Require,ID }

36- follow(F\_name)={.}

Parse Tree

Shape

Description automatically generated with medium confidence

AST

Shape

Description automatically generated with medium confidence