CS 419 Compiler Projects Form

1) Instructions to be Followed (for the Hard Copy):

I. Fill page **NO 4** with the required Fields:

I. Project Idea: The **Idea** will be assigned to you .

II. Team Members NO#: Number of team members **7**

III. 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:

I. **Project Policy:** affects all projects members including team leader.

II. **Member Policy:** affects a member of project's members.

3) General Notes:

I. Total grade of Project is 15

II. **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**.**

III. Once you Registered, **NO modifications** will be done.

IV. Allowed only on registration for team in form, duplication will got **-2 Project Policy.**

V. Each group will be assigned **an Idea, ID and time slot** for **discussion**.

VI. 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:

I. **Copied Code** will be got ZERO Without Discussion.
II. 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**.
III. Each team member must have **a complete knowledge** about the whole project
IV. **Evaluation** will be **Individual Evaluation** not project Evaluation.
V. **-2 Project Policy** will be applied in case of being late for assigned discussion time slot
VI. **NO discussion will be repeated under any circumstances.**
VII. 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**).
VIII. Discussion Day will be **sent later** .

5) Notes about Implementation:

I. **.Net or PHP** are only allowed.
II. The project must be a Web (use latest technologies).
III. Your code must be uploaded to github before discussion.
IV. **- 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).
V. 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).

VI. 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.

VII. 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.

VIII. – 3 Project policy will be applied if the content of the file that is mentioned in point V is opened or viewed.

5) Notes about Discussion Testing:

- There will be two types of Testing :
  I. **White Box Testing:** This will be from **Editor**.
  II. **Black Box Testing:** This will be from "**Browse**" Button

Thanks,

CS 419 Compiler

Project Form

Project Idea:

……

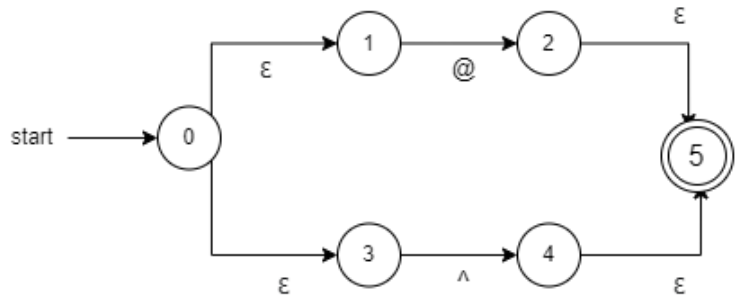
Team Members NO#: ……..

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 عبدالله	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	

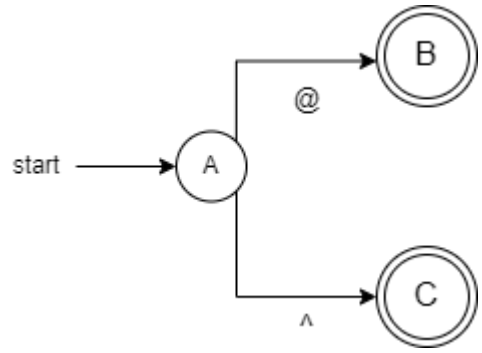
Regex:

Start-> (@|^)

NFA:



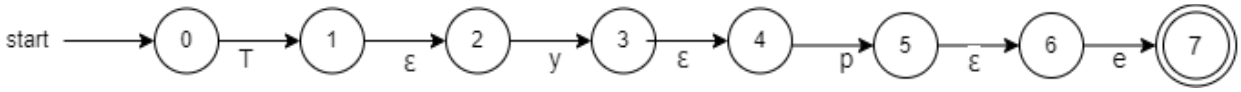
DFA:



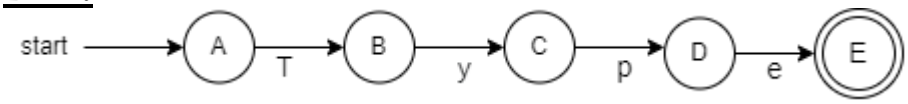
Regex:

Class -> (Type)

NFA:



DFA:

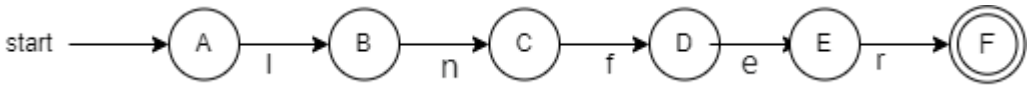


Regex:

Inheritance -> (Infer)

NFA:

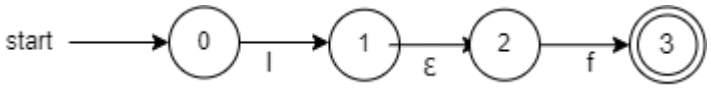
DFA:



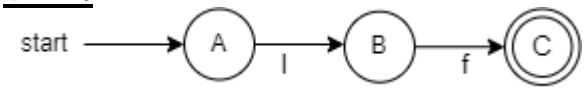
Regex:

If -> (If)

NFA:



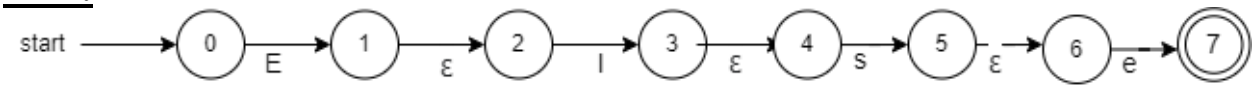
DFA:



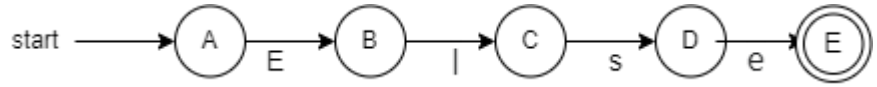
Regex:

Else-> (Else)

NFA:



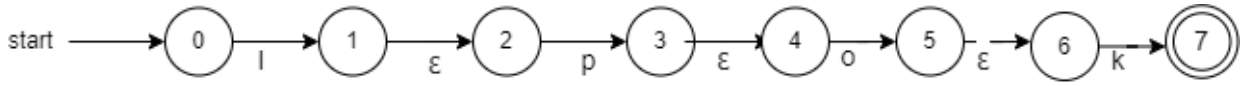
DFA:



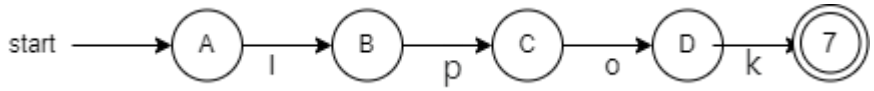
Regex:

Integer -> (Ipok)

NFA:



DFA:

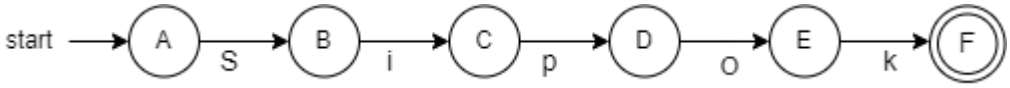


Regex:

SInteger-> (Sipok)

NFA:

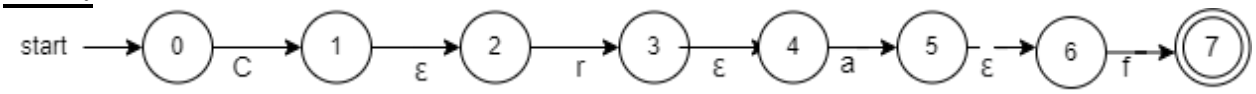
DFA:



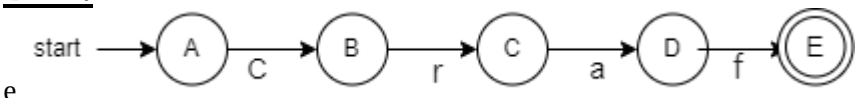
Regex:

Character-> (Craf)

NFA:



DFA:

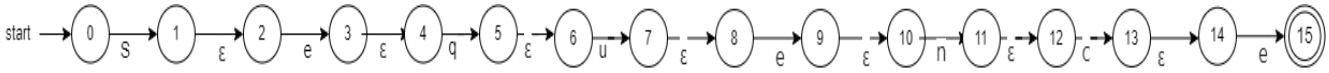


e

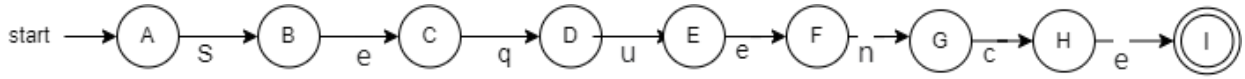
Regex:

String-> (Sequence)

NFA:



DFA:

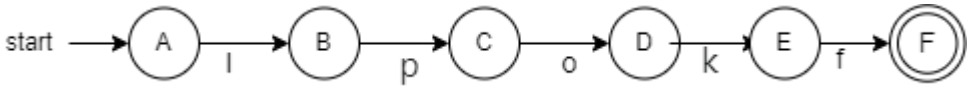


Regex:

Float -> (Ipokf)

NFA:

DFA:

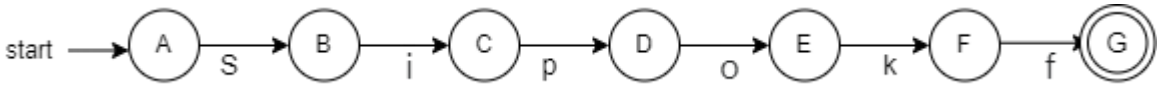


Regex:

Sfloat -> (Sipokf)

NFA:

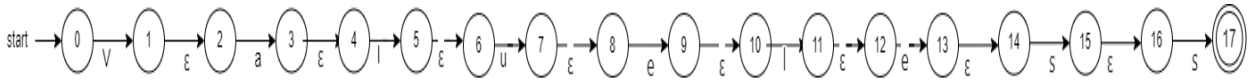
DFA:



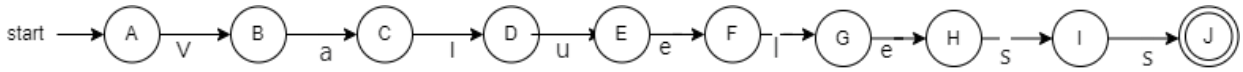
Regex:

Void -> (Valueless)

NFA:



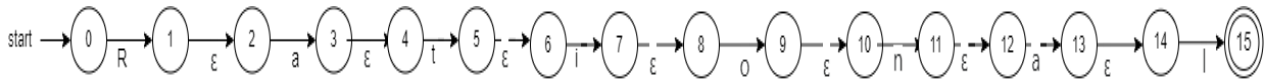
DFA:



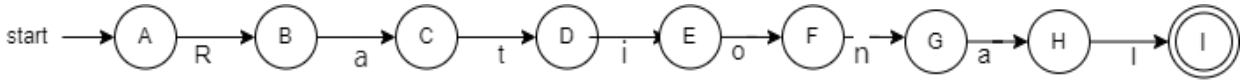
Regex:

Boolean -> (Rational)

NFA:



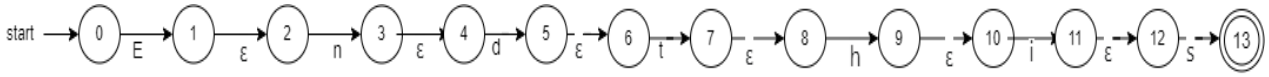
DFA:



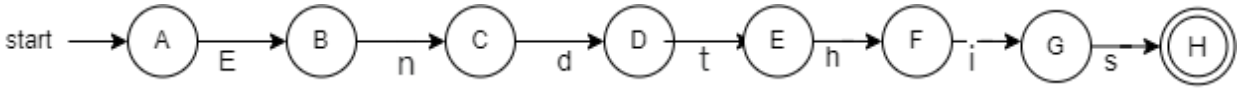
Regex:

Break-> (Endthis)

NFA:



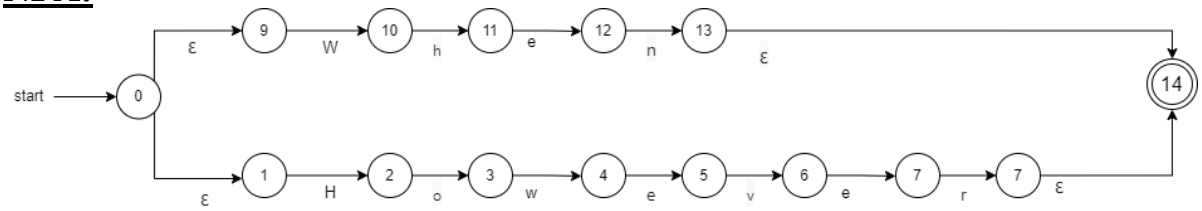
DFA:



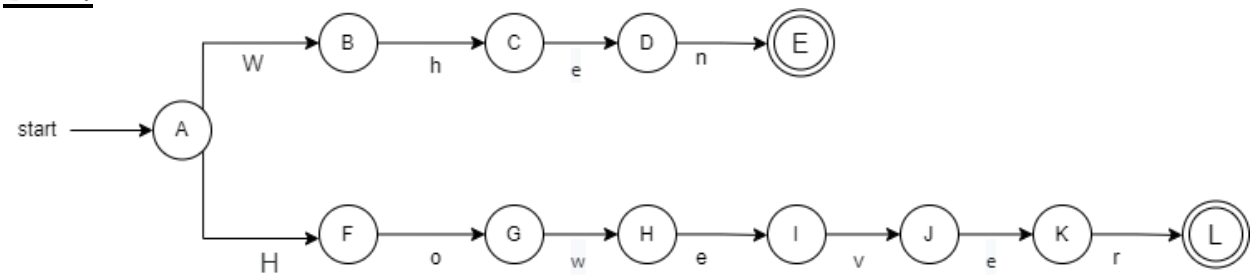
Regex:

Loop -> (However|When)

NFA:



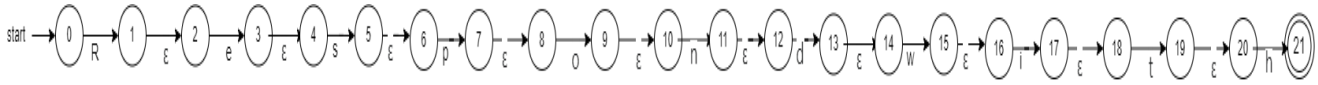
DFA:



Regex:

Return -> (Respondwith)

NFA:

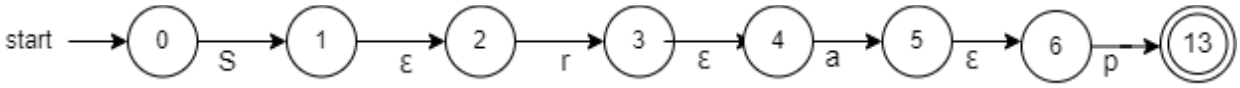


DFA:

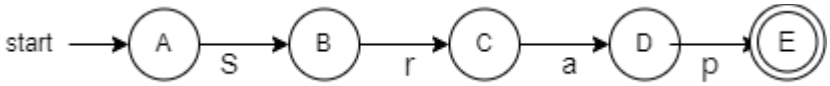
Regex:

Struct -> (Srap)

NFA:



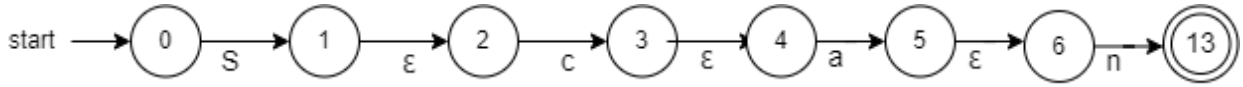
DFA:



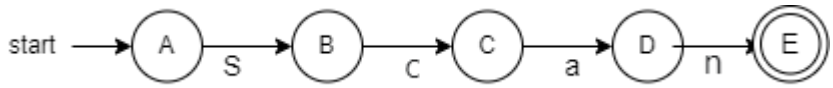
Regex:

Switch -> (Scan)

NFA:



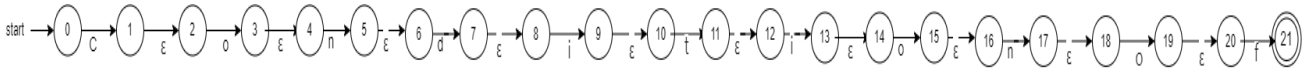
DFA:



Regex:

Case -> (Conditionof)

NFA:

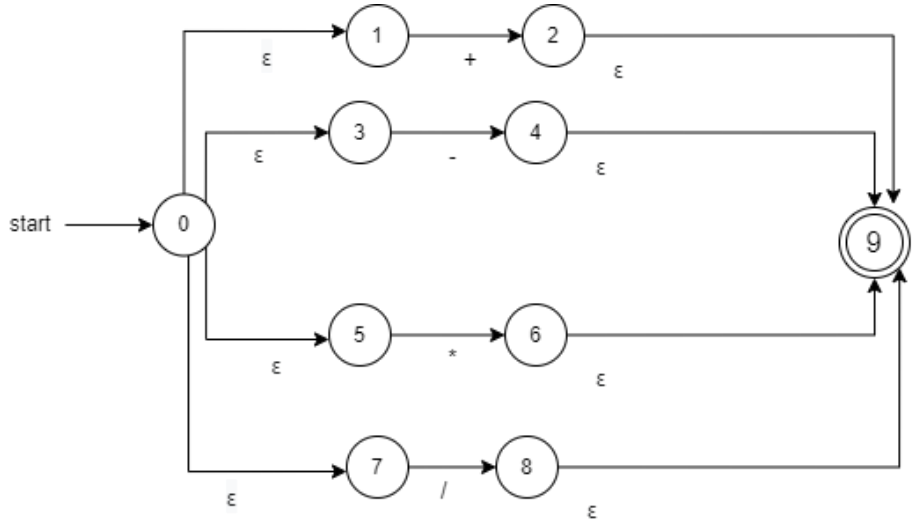


DFA:

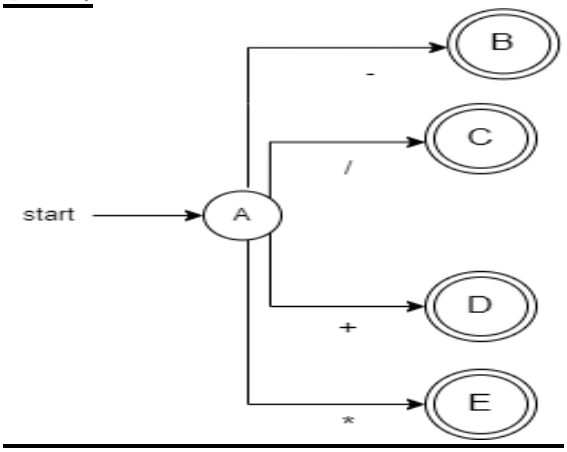
Regex:

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

NFA:



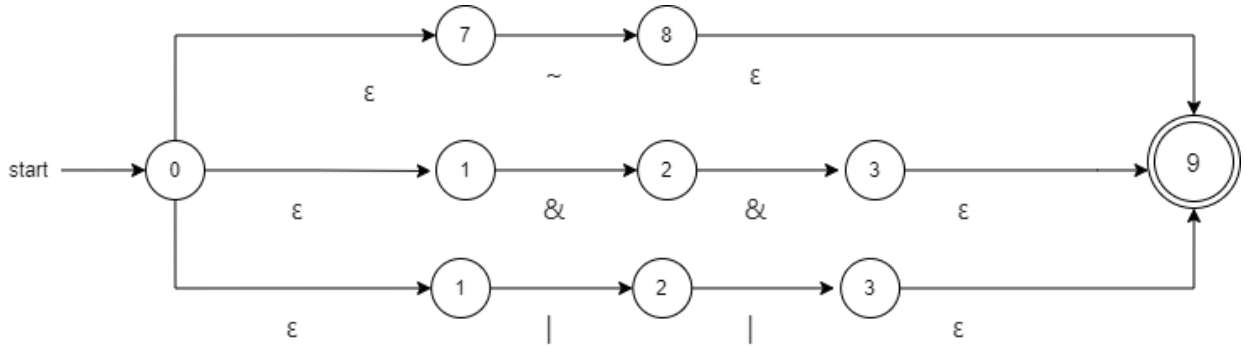
DFA:



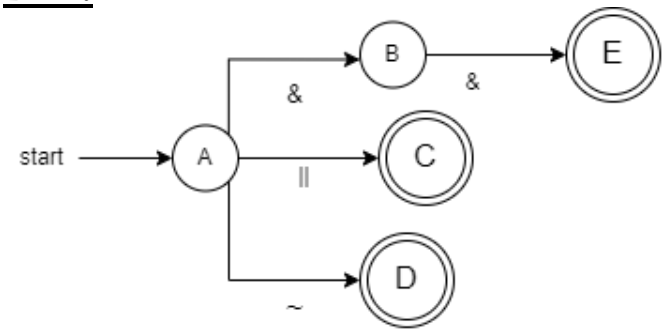
Regex:

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

NFA:



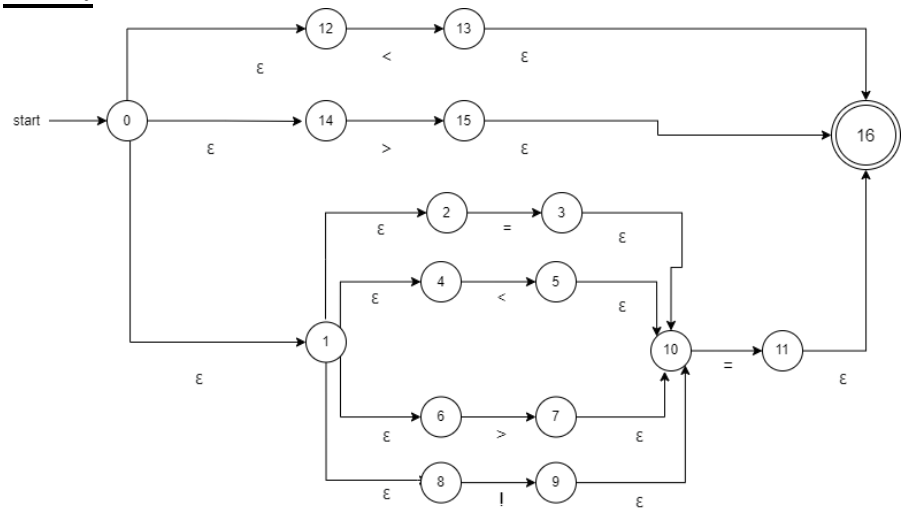
DFA:



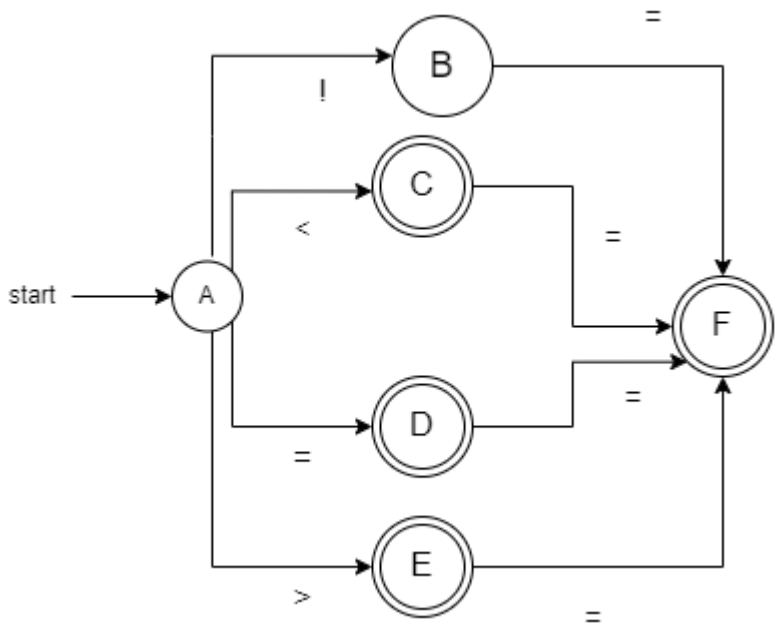
Regex:

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

NFA:



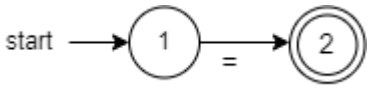
DFA:



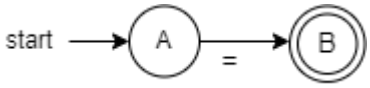
Regex:

Assignment -> (=)

NFA:



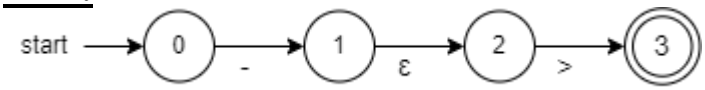
DFA:



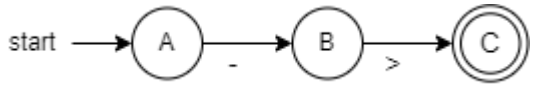
Regex:

Access -> (->)

NFA:



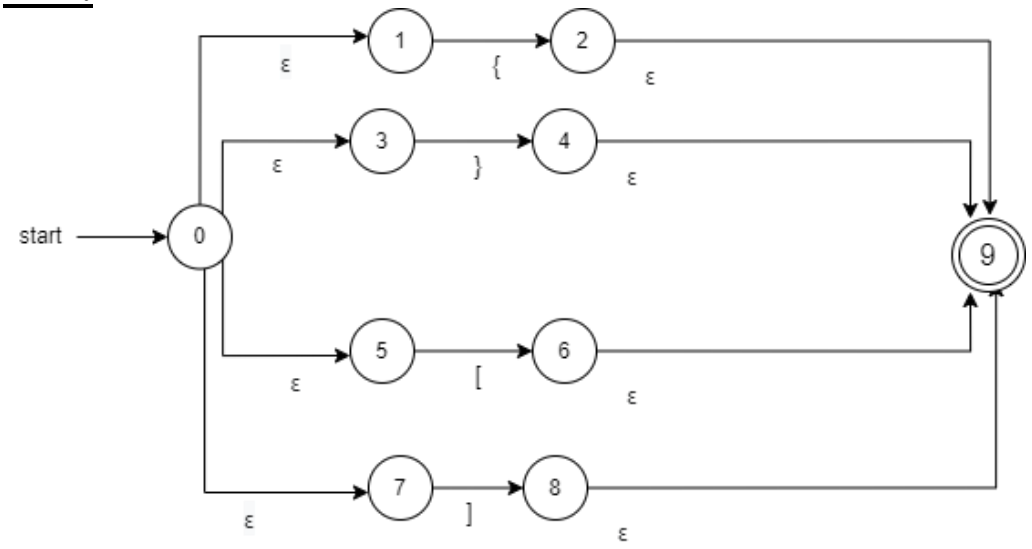
DFA:



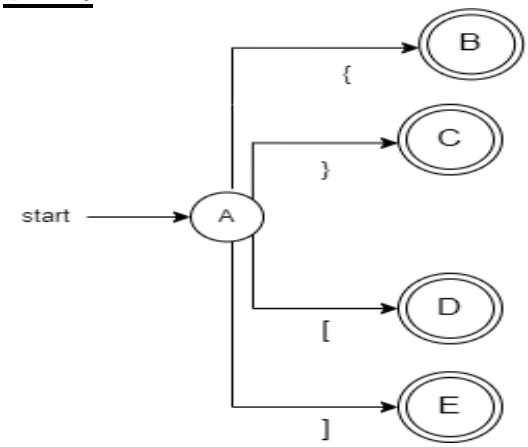
Regex:

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

NFA:



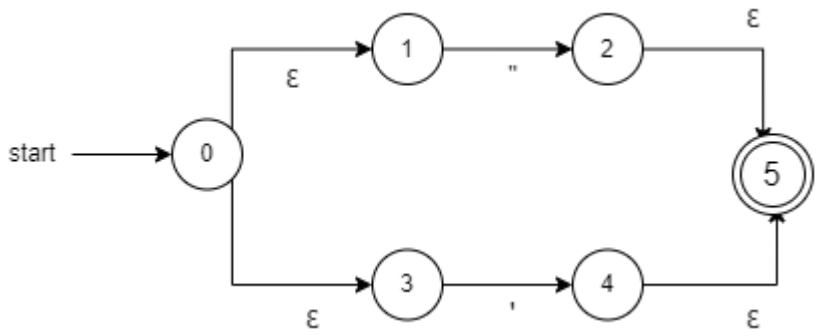
DFA:



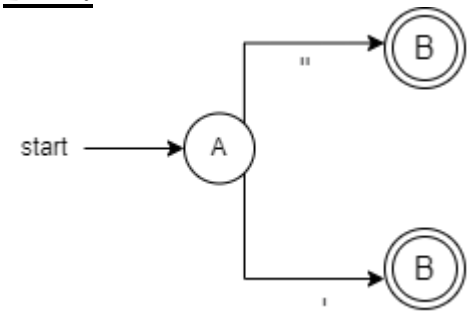
Regex:

Quotation -> (“|’)

NFA:



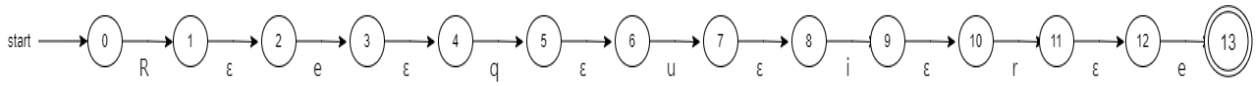
DFA:



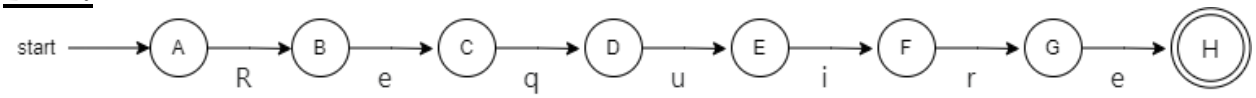
Regex:

Inclusion -> (Require)

NFA:



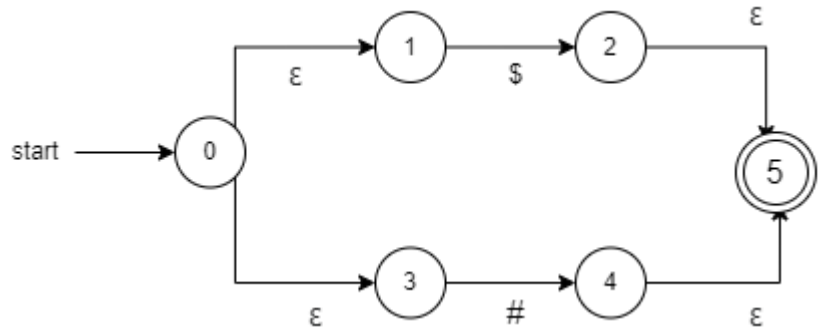
DFA:



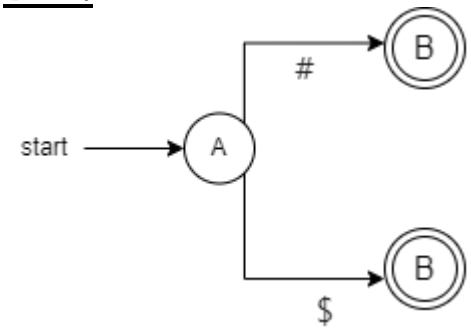
Regex:

End -> ($|#)

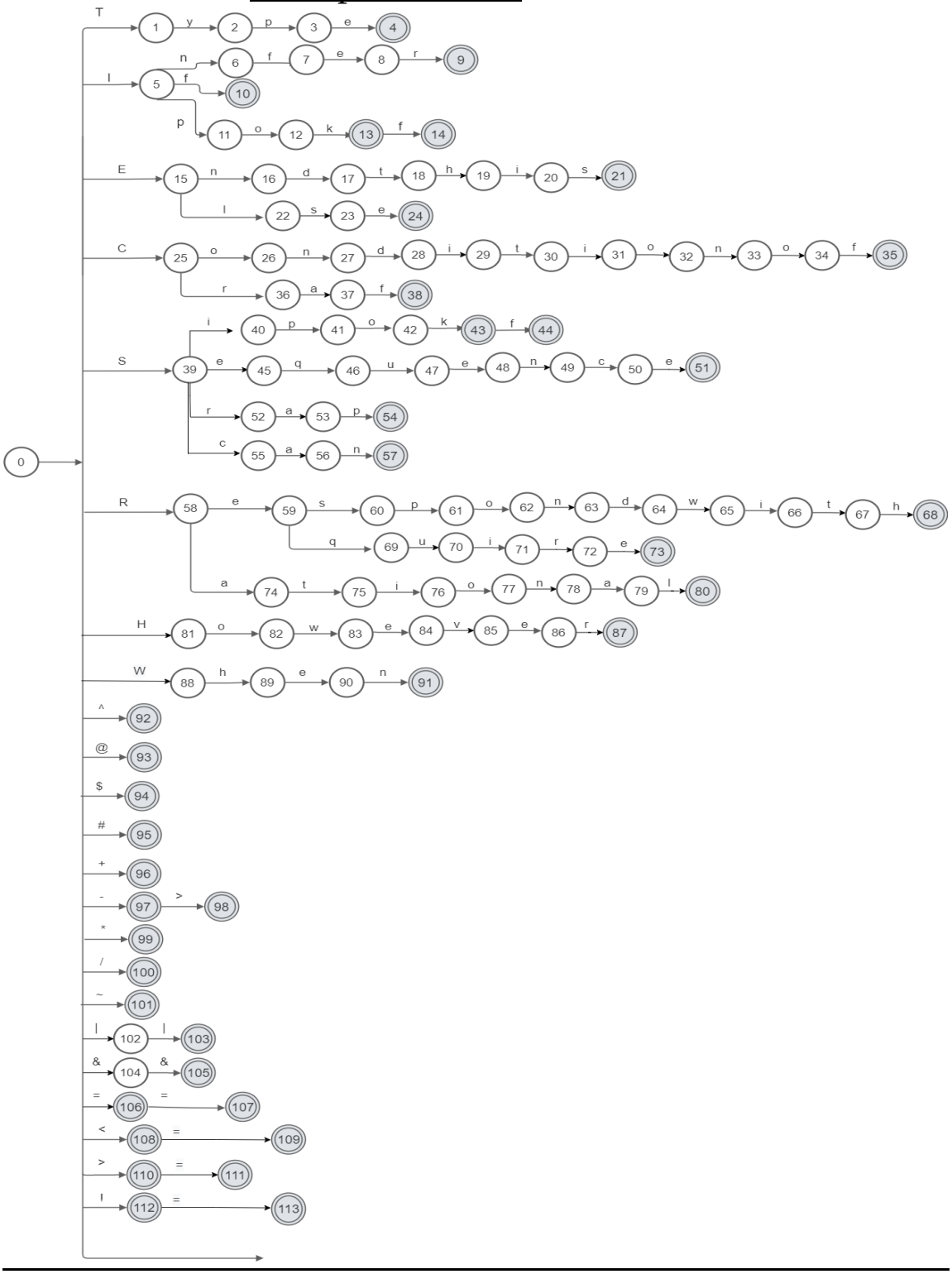
NFA:



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}

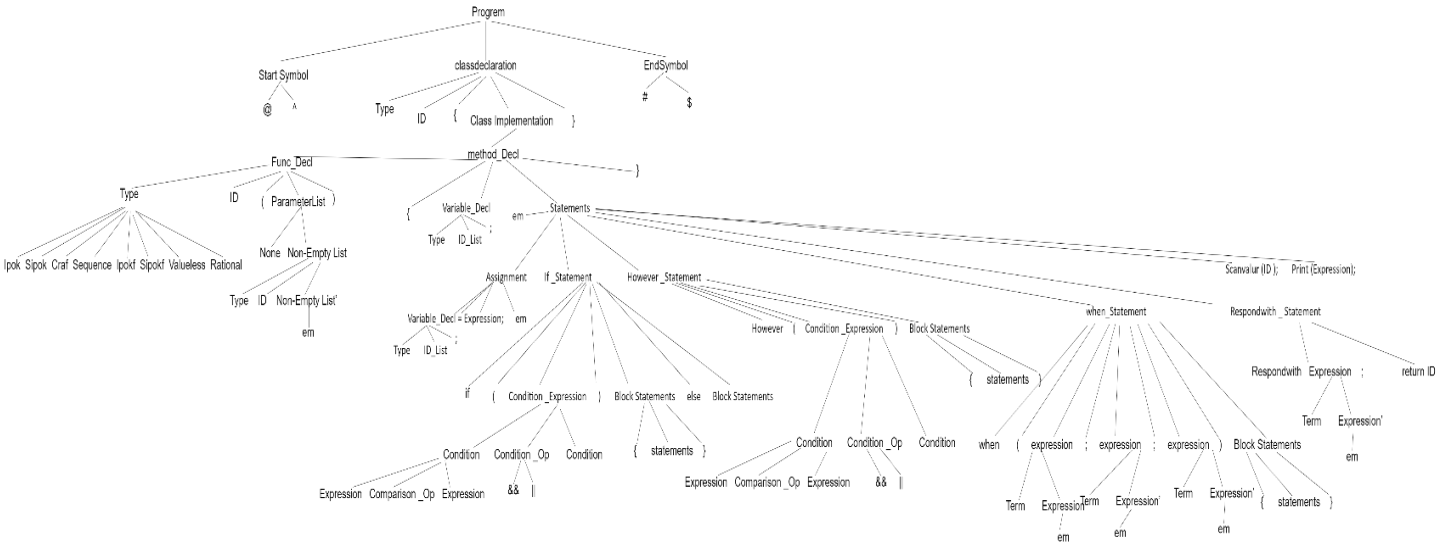
36-First(F_name) = {STR}

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



AST