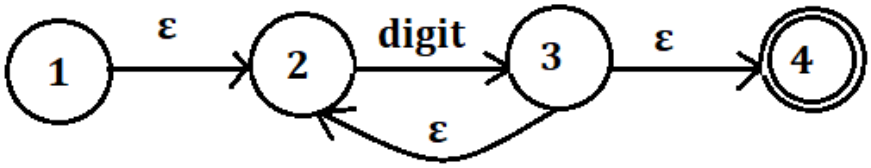
Compiler paroject#3

Phase 1 Scanner

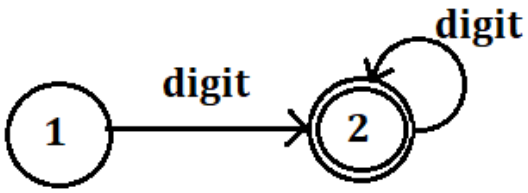
RE(digit) = ([0-9])*

blob:file:///a95ff0ca-a814-439e-b07b-311c3061570f

NFA



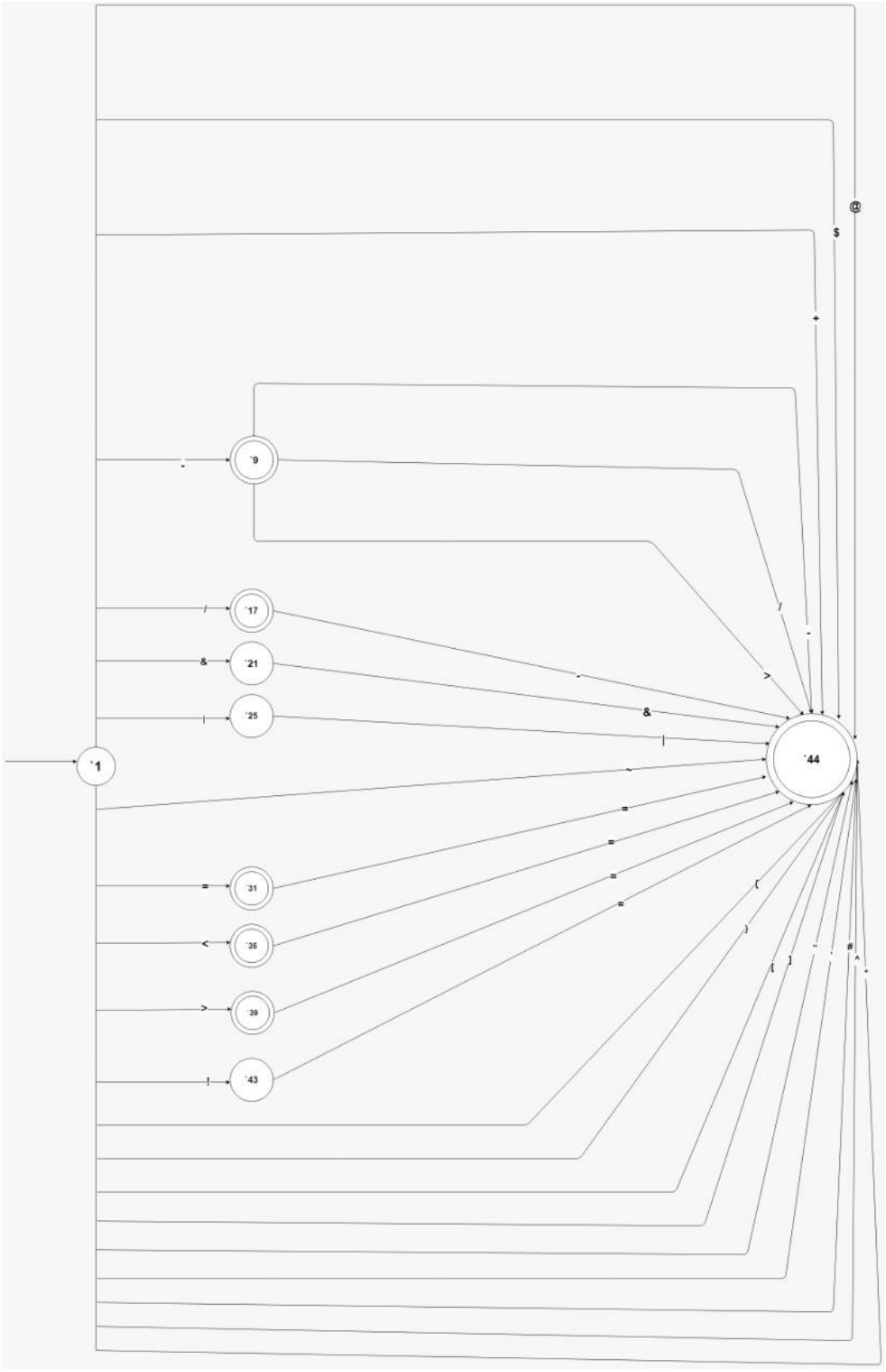
DFA

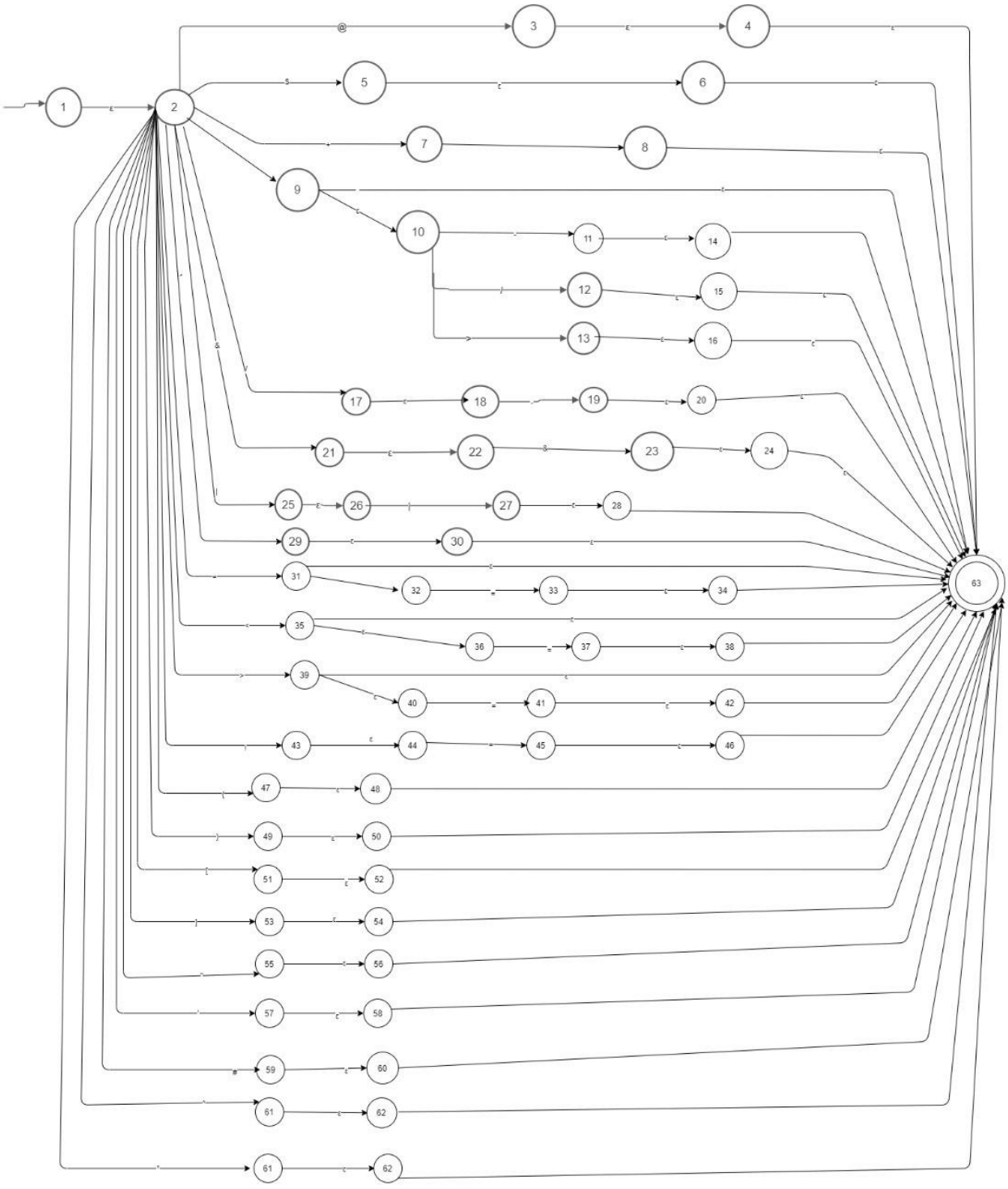


Transition table

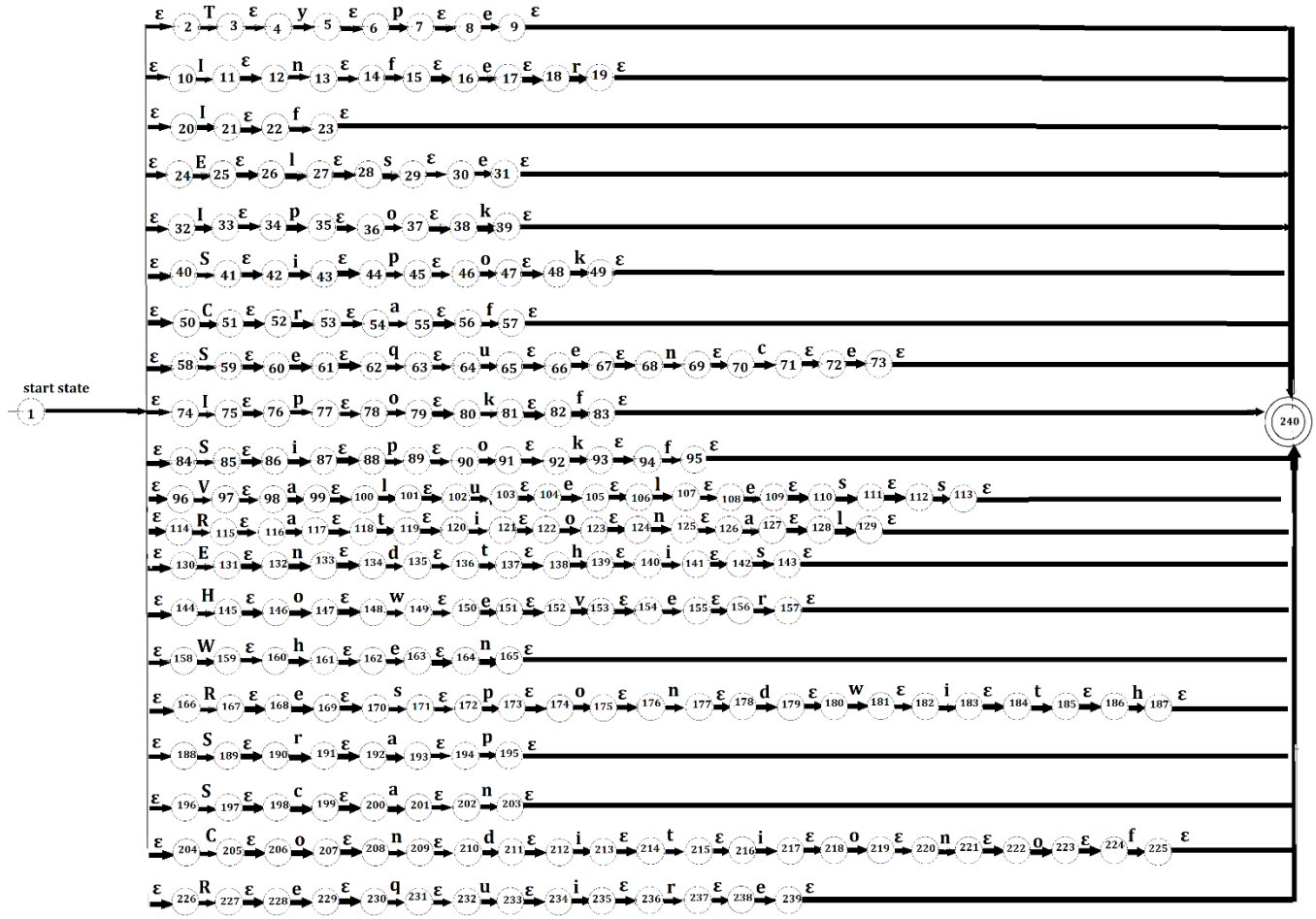
state	Digit	Accepting
1	2	NO
2	2	YES

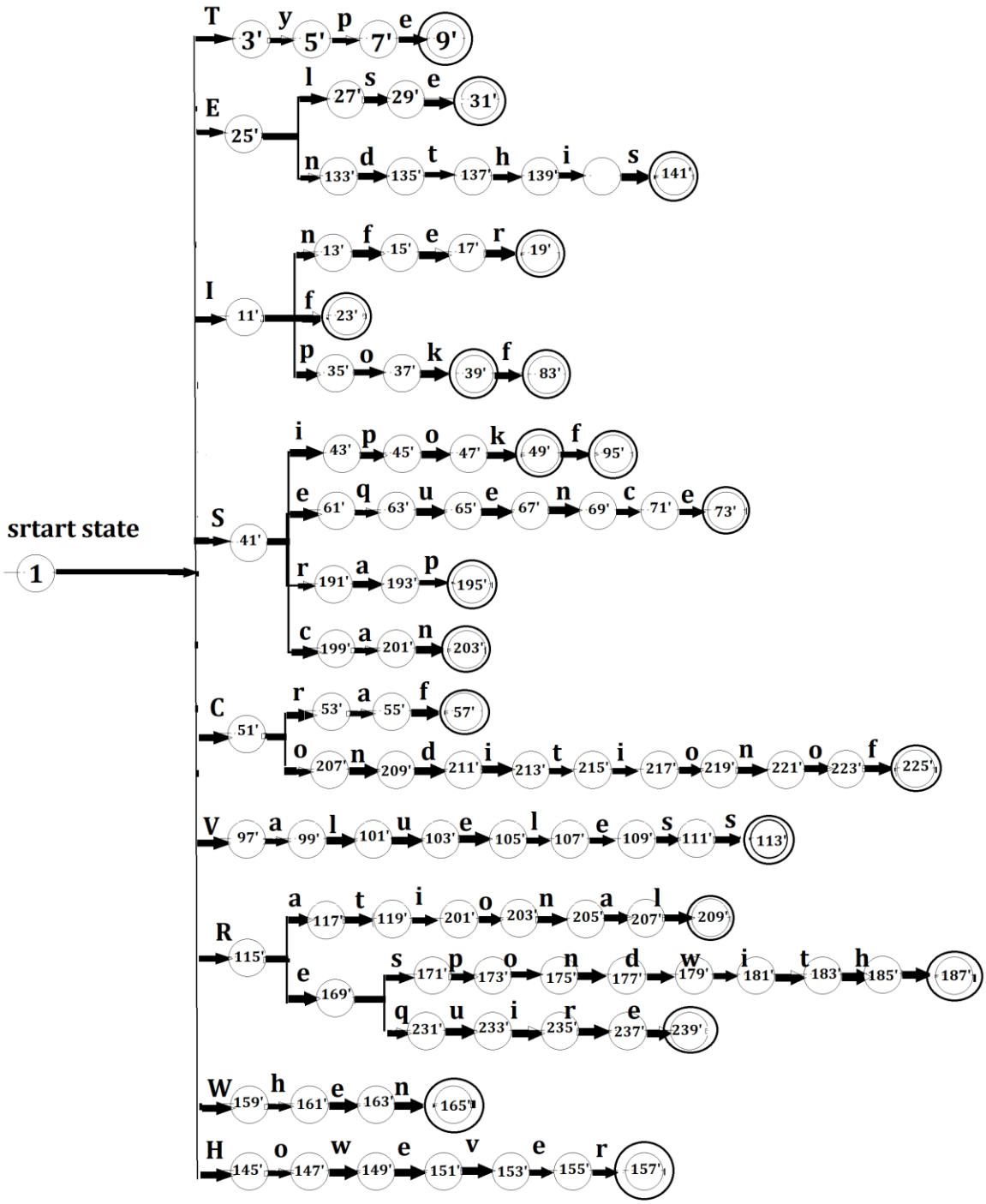
RE(symbols) = (@|$|+|-|*| /|&&| || | ~| ==| <| >| !=| <=| >=| =| ->| {| }| [|]| '| "| /-| -/ |--| #| ^).


`1
`9
`17
`21
`25
`31
`35
`39
`43
`44
-
/
&
|
=
<
>
!
@
$
+
*
(
)
{
}
#
^



RE(KW) = (TYype| Infer | If | Else | Ipok | Sipok | Craf | Sequence | Ipokf | Sipkf | Valueless | Rational | Endthis | However | When | Respondwith | Srap | Scan | Conditionof | Require)

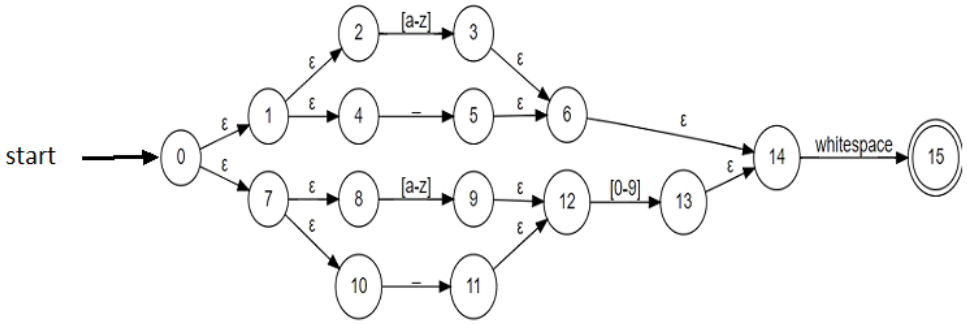

start state
1
240


srtart state
1
T
3'
y
5'
p
7'
e
9'
E
25'
l
27'
s
29'
e
31'
n
133'
d
135'
t
137'
h
139'
i
s
141'
I
11'
n
13'
f
15'
e
17'
r
19'
f
23'
p
35'
o
37'
k
39'
f
83'
S
41'
i
43'
p
45'
o
47'
k
49'
f
95'
e
61'
q
63'
u
65'
e
67'
n
69'
c
71'
e
73'
r
191'
a
193'
p
195'
c
199'
a
201'
n
203'
C
51'
r
53'
a
55'
f
57'
o
207'
n
209'
d
211'
i
213'
t
215'
i
217'
o
219'
n
221'
o
223'
f
225'
V
97'
a
99'
l
101'
u
103'
e
105'
l
107'
e
109'
s
111'
s
113'
R
115'
a
117'
t
119'
i
201'
o
203'
n
205'
a
207'
l
209'
e
169'
s
171'
p
173'
o
175'
n
177'
d
179'
w
181'
i
183'
t
185'
h
187'
q
231'
u
233'
i
235'
r
237'
e
239'
W
159'
h
161'
e
163'
n
165'
H
145'
o
147'
w
149'
e
151'
v
153'
e
155'
r
157'

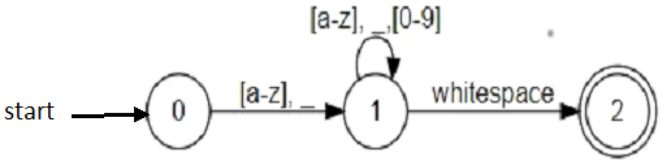
Identifier:

RE → (([a-z] | _)* | (([a-z] | _)[0-9])*)whitespace

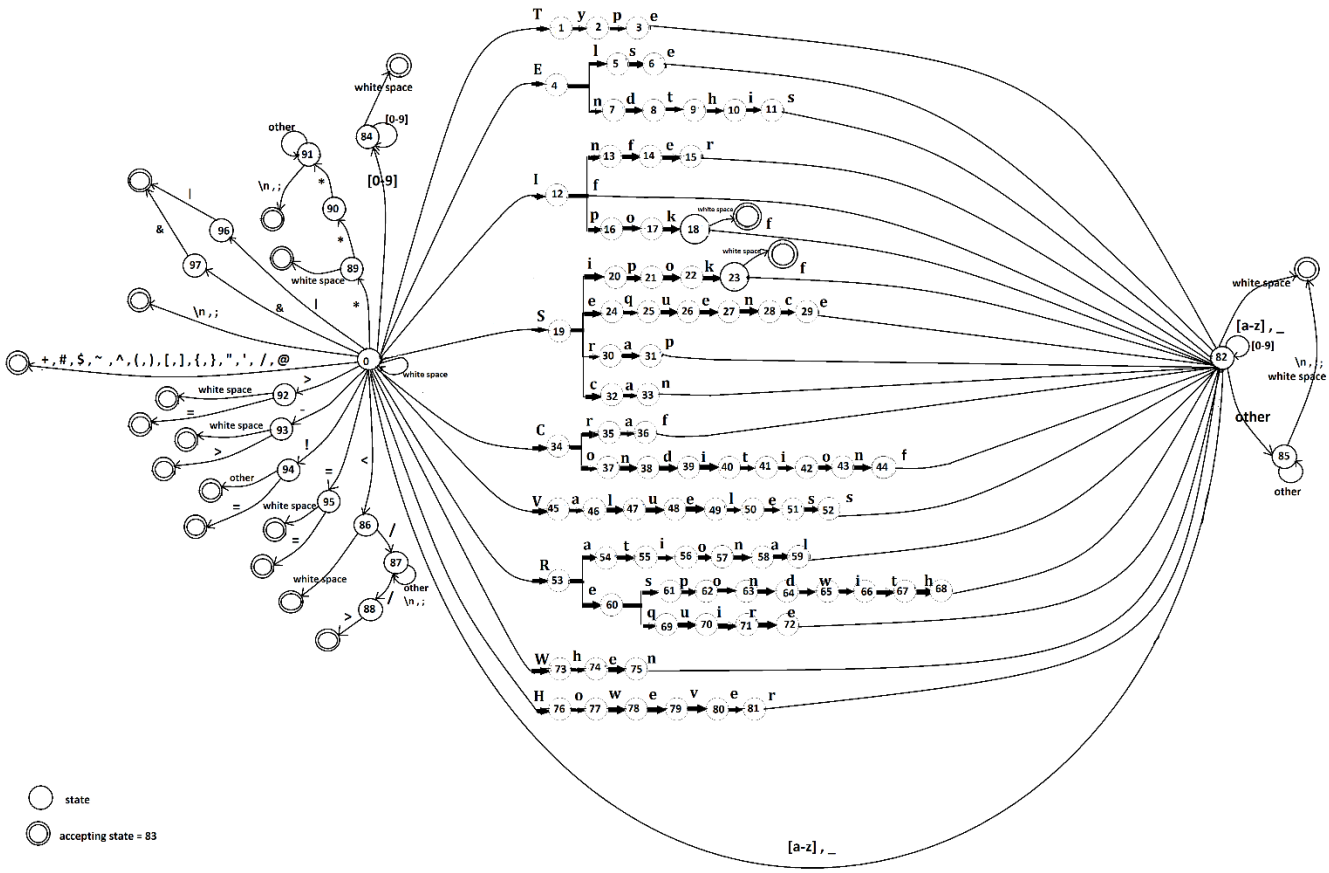
NFA



DFA



Last DFA:



Transition diagram:

https://docs.google.com/spreadsheets/d/1OSxOYvQyQrBnU_ygXjNBk26MJrCCI0ko/edit?usp=sharing&ouid=105254318349782653163&rtpof=true&sd=true

Phase 2 Parser

First:

1- First(program)=First(Start_Symbols)={@,^ }
2- First(Start_Symbols)={@,^ }
3- First(End_Symbols)={$,#}
4- First(ClassDeclaration) = First(Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational}
5- First(Class_Implementation) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, < , *, require, ID, em}
6- First(Method_Decl) = First(Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational}
7- First(Func Decl) = First(Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational}
8- First(Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational}
9- First(ParameterList) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, None, em}
10- First(Non-Empty List) = First(Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational}
10.1- First(Non-Empty List') = {**,** , em}
11- First (Variable_Decl) = First (Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, em}
12- First (ID_List) = {ID}
12.1-First (ID_List') = {**,** , em}
13- First (Statements) = First (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print, em}
14- First (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print}
15- First (Assignment) = First (Variable_Decl) = First (Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, ID, Number}
16- First (Func _Call) ={ID}
17- First (Argument_List)={ ID, Number ,em}
18- First (NonEmpty_Argument_List) ={ID, Number}
18.1-First (NonEmpty_Argument_List')={, , em}
19-First (Block Statements)= {{}
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}
28- First(Endthis _Statement)={ Endthis}
29- First(Expression)={ ID, Number }
29.1- First(Expression')={+, -, em }
30- First(Add_Op)={ +, -}
31- First(Term)={ ID, Number }
31.1- 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}

Follow:

1- Follow(program)={$}
2- Follow(Start_Symbols)= First(ClassDeclaration) = First(Type) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational}
3- Follow(End_Symbols)= Follow(program)={$ }
4- Follow(ClassDeclaration) = First(End_Symbols)={$,#}
5- Follow(Class_Implementation) = { }}
6- Follow(Method_Decl) = First(Class_Implementation) ∪ Follow(Class_Implementation)= { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, < , *, require, ID, }}
7- Follow(Func Decl) = Follow(Type) ={; , {}
8- Follow(Type) = {ID}
9- Follow(ParameterList) = {)}
10- Follow(Non-Empty List) = Follow(ParameterList) = {)}
10.1 - Follow(Non-Empty List') = Follow(Non-Empty List) = Follow(ParameterList) = {)}
11- Follow (Variable_Decl) ={ Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, < , *, require, ID, }, If, when, however, respondwith, endthis, Scanvalur, Print,=}
12- Follow (ID_List) = {; , [}
12.1-Follow (ID_List') = Follow (ID_List) = {; , [}
13- Follow (Statements) ={}}
14- Follow (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}}
15- Follow (Assignment) = Follow (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}}
16- Follow (Func _Call) = First(Class_Implementation) ∪ Follow(Class_Implementation)= { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, < , *, require, ID, }}
17- Follow (Argument_List)={)}
18- Follow (NonEmpty_Argument_List) = Follow (Argument_List)={)}
18.1-Follow (NonEmpty_Argument_List')= Follow (NonEmpty_Argument_List) = Follow (Argument_List)={)}
19-Follow (Block Statements)= { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}, else}
20-Follow (If _Statement)= Follow (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}}
21-Follow (Condition _Expression)={) }
22- Follow (Condition _Op)= First(Expression)={ ID, Number }

23- Follow (Condition)= First (Condition _Op)={ && , ||}
24-Follow(Comparison _Op)= First(Term)={ ID, Number }
25- Follow(However _Statement)= Follow (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}}
26- Follow(when _Statement)= Follow (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}}
27- Follow(Respondwith _Statement)= Follow (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}}
28- Follow(Endthis _Statement)= Follow (Statement) = { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, If, when, however, respondwith, endthis, Scanvalur, Print,}}
29- Follow(Expression)={), ; , , , == , != , > , >= , < , <=}
29.1- Follow(Expression')= Follow(Expression)={), ; , , , == , != , > , >= , < , <=}
30- Follow(Add_Op)= First(Term)={ ID, Number }
31- Follow(Term)= {+, -,), ; , , , == , != , > , >= , < , <=}
31.1- Follow(Term')= Follow(Term)= {+, -,), ; , , , == , != , > , >= , < , <=}
32-- Follow(Mul_Op)= First(Factor)={ ID, Number }
33- Follow(Factor)={ *, /,+, -,), ; , , , == , != , > , >= , < , <=}
34- Follow(Comment)= First(Class_Implementation) ∪ Follow(Class_Implementation)= { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, < , *, require, ID, }}
35- Follow(Require_command)= First(Class_Implementation) ∪ Follow(Class_Implementation)= { Ipok, Sipok, Craf, Sequence, Ipokf, Sipokf , Valueless, Rational, < , *, require, ID, }}
36- Follow(F_name)={.}

Parsing table

https://docs.google.com/spreadsheets/d/1hW558CJvmnxuZPnhzk8tfkn9BV1ZXxH4/edit?usp=sharing&ouid=105254318349782653163&rtpof=true&sd=true

Parse Tree

```
1- @ Type Person{
2- Rational G() {
3- int frt=5;
4- when (in
counter<num){
5- int reg3=reg3-1; } }} $
```