

CS 419 Compiler

Project Form

Project Idea:

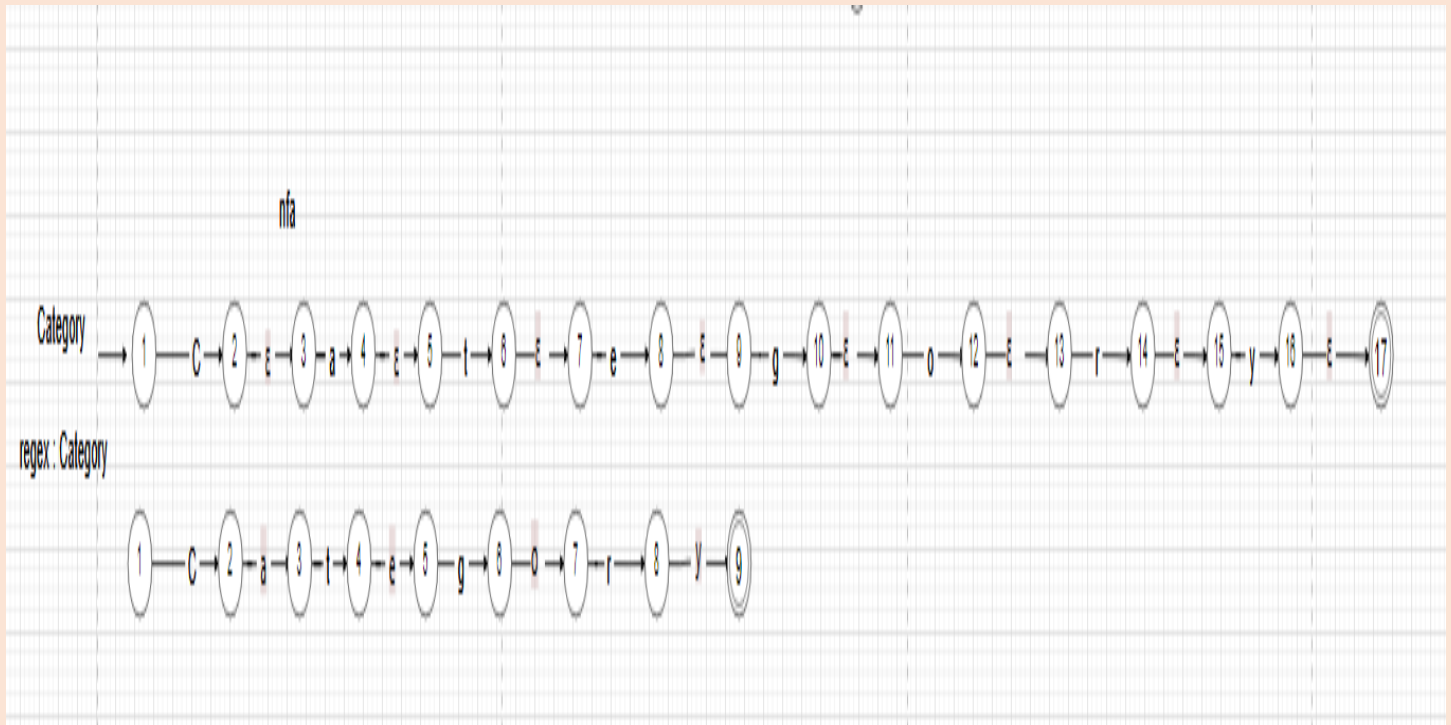
Project Idea #2

Team Members NO#: 7

ID	Name	Level& Department	Section(Day- from-to)	Role (Lead/Member)	Grade
201900754	محمود احمد صلاح علي	3		leader	
201900728	محمد محمود الدمرداش لاشين	3			
20180441	كيرلس ميثيل سعيد عطا الله	4			
20150644	يوسف محمود محمد رجب	4			
20180436	كريم محمد يوسف شطا	4			
20160067	اسلام سيد صلاح	4			
20170409	محمد احمد عياد	4			

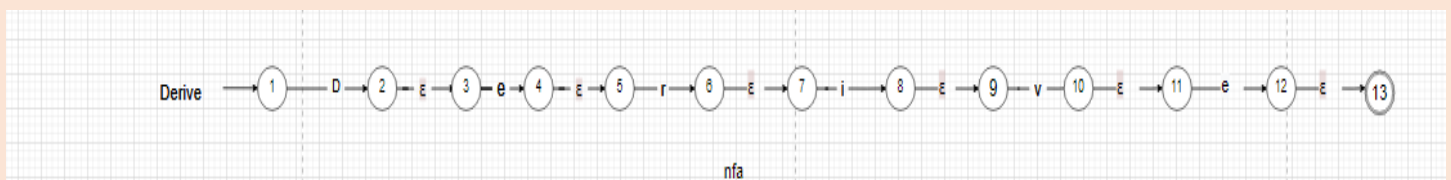
**Regular Expression, Finite automata and Conversion from RegX
to NFA, NFA to DFA**

1-Category:

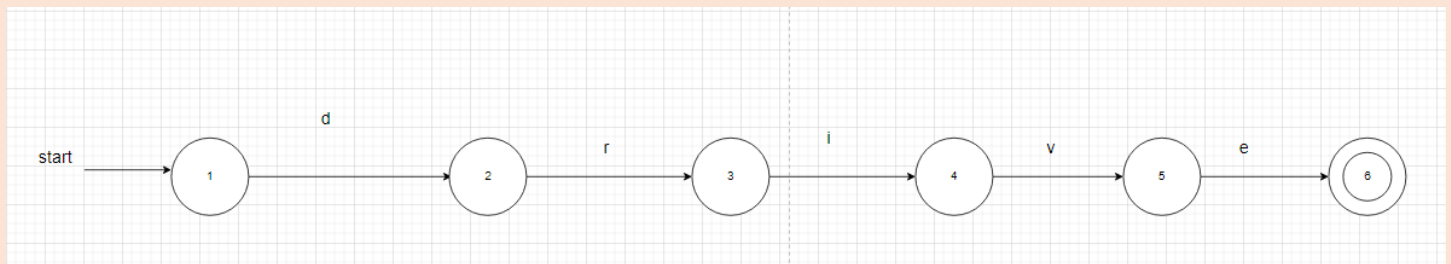


2-Derive:

N f a :

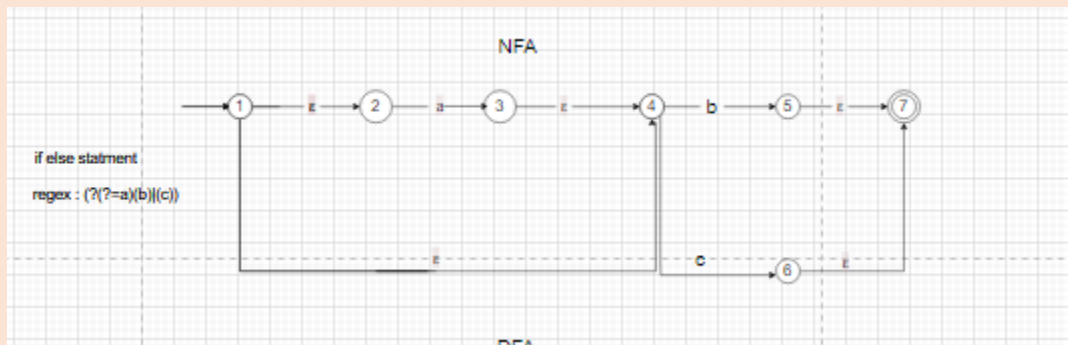


D f a :

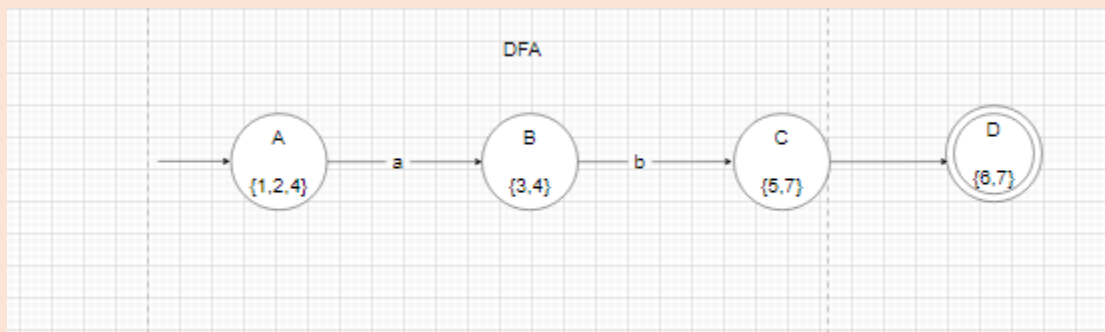


3-Else if:

N f a :

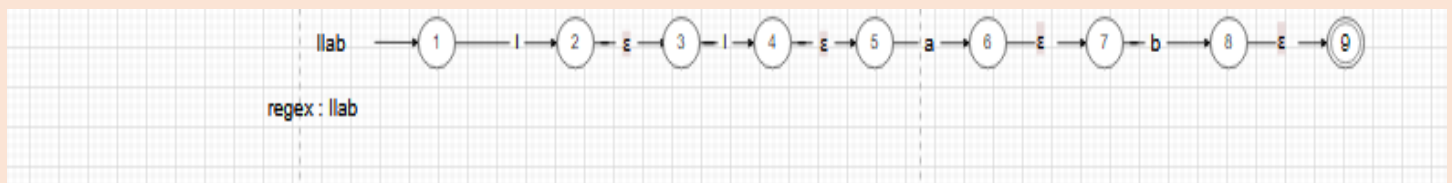


Dfa :

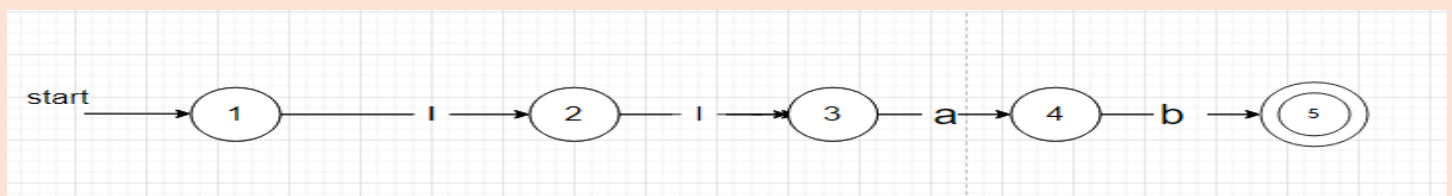


4-llap:

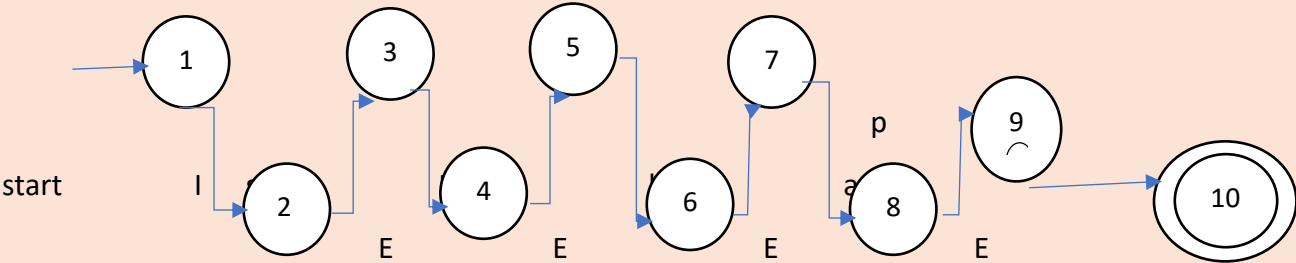
N f a :



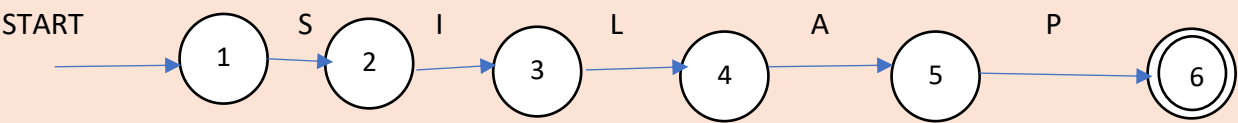
D f a :



5-Silap:



D F A:

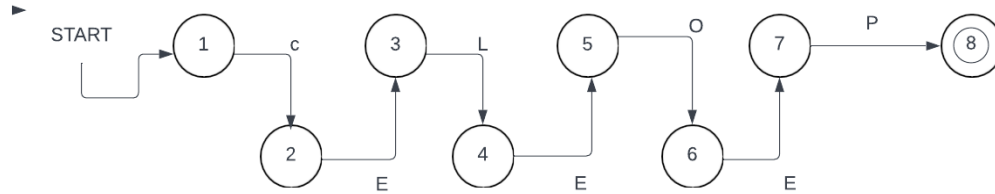


TRANSACTION TABLE:

	S	I	L	A	P	
1	2					
2		3				
3			4			
4				5		
5					6	
6						

6-CLOP:

NFA:



DFA:

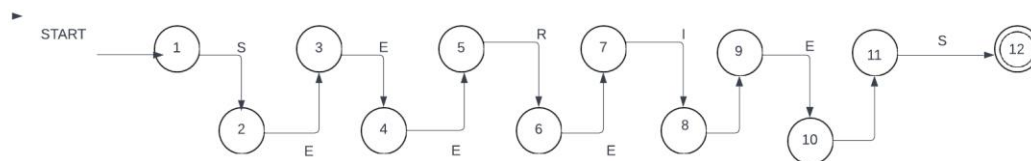


TRANZACTION TABLE:

	C	L	O	P	
1	2				
2		3			
3			4		
4				5	
5					

7-SERIES:

NFA :



DFA:

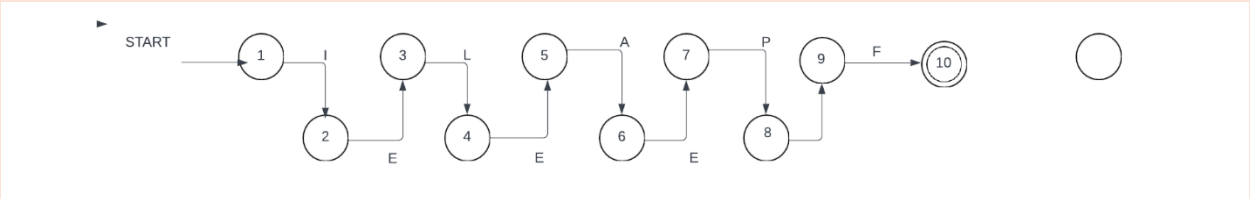


TRANSACTION TABLE:

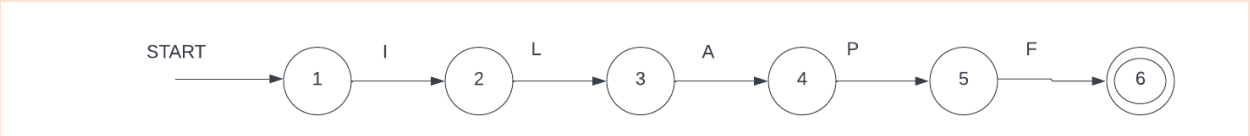
	S	E	R	I	E	S
1	2					
2		3				
3			4			
4				5		
5					6	
6						7
7						

8-ILAPF:

N F A :



D F A:

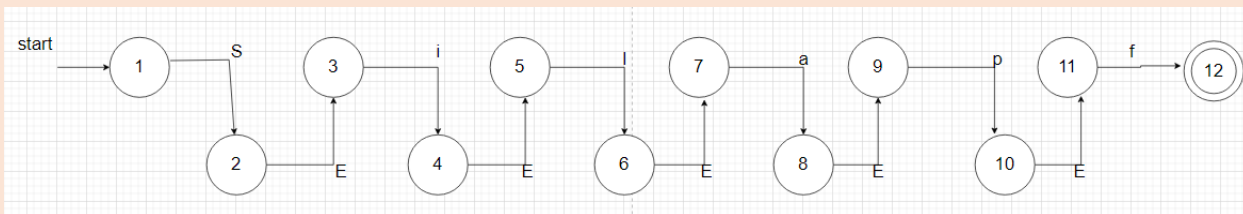


TRANZACTION TABLE:

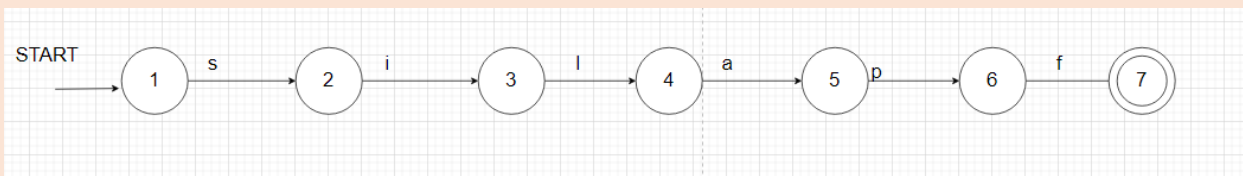
	I	L	A	P	F	
1	2					
2		3				
3			4			
4				5		
5					6	
6						

9-Silapf:

NFA:



Dfa:

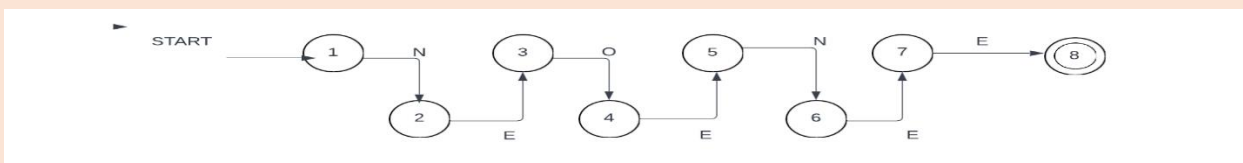


TRANSACTION TABLE :

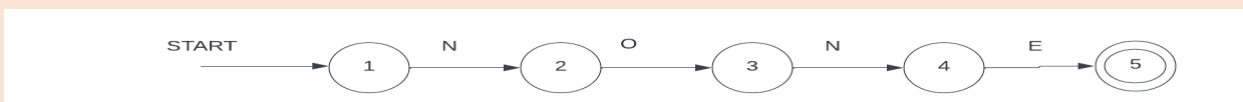
	S	I	L	A	P	F
1	2					
2		3				
3			4			
4				5		
5					6	
6						7
7						
8						

10-NONE:

NFA



DFA:

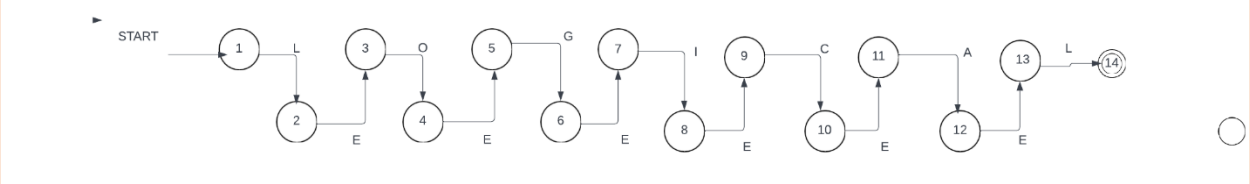


TRANZACTION TABLE :

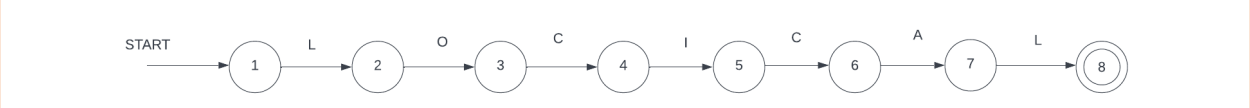
	N	O	E	
1	2			
2		3		
3	4			
4			5	
5				

11-LOGICAL:

N F A:



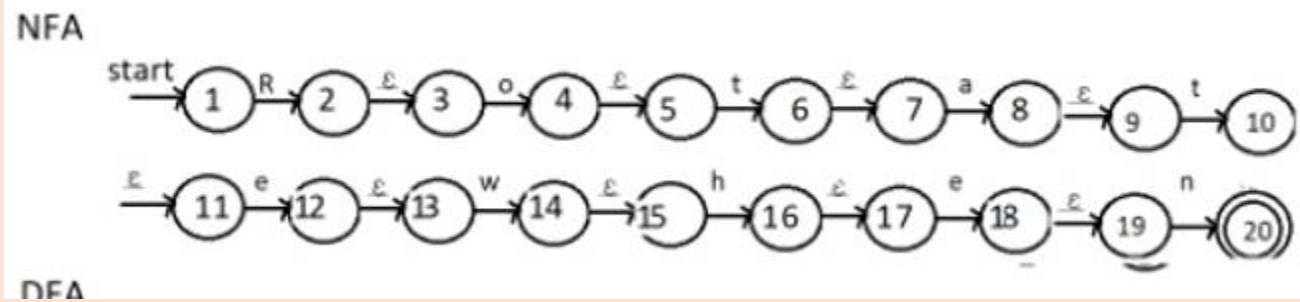
D F A:



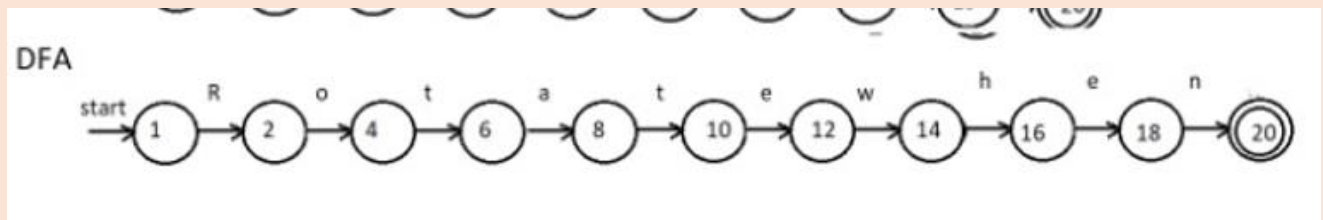
TRANSACTION TABLE :

	L	O	G	I	C	A
1	2					
2		3				
3			4			
4				5		
5					6	
6						7
7	8					
8						

12-Rotate when:



D f a :

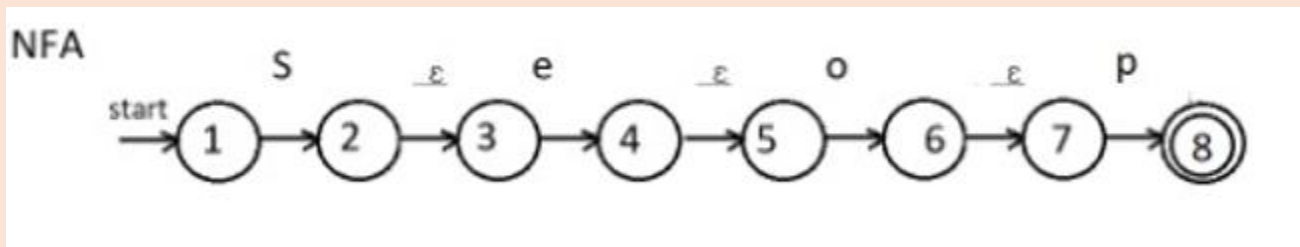


TRANSACTION TABLE :

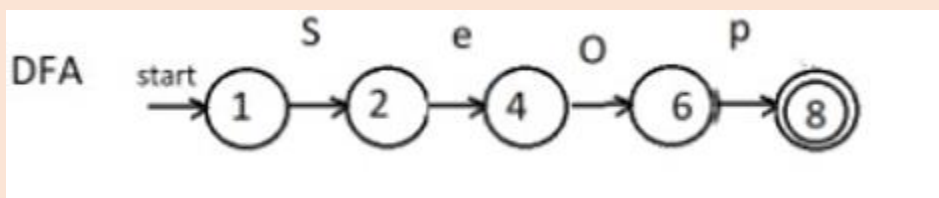
[illegible]

13-Seop:

N f a:



D f a:

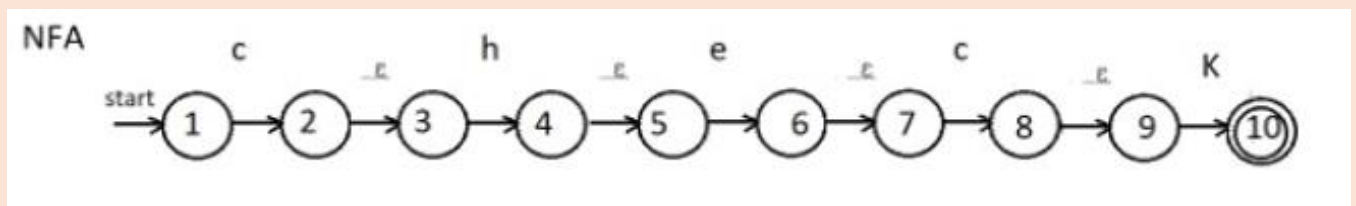


TRANSACTION TABLE :

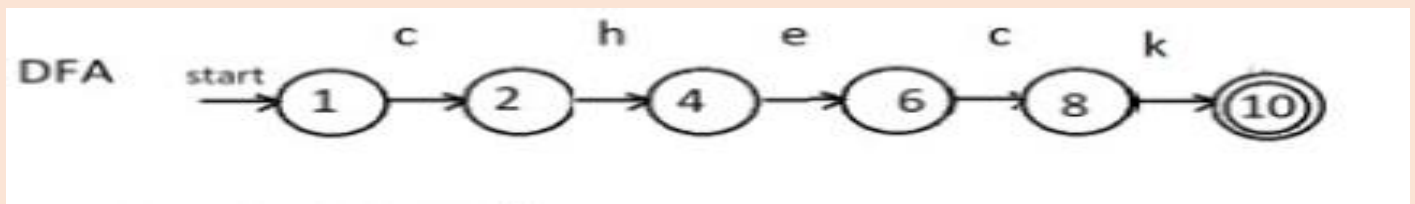
	s	e	o	p
1	2'			
2		4'		
4			6'	
6				8'
8				

14-Check:

N f a:



D f a:

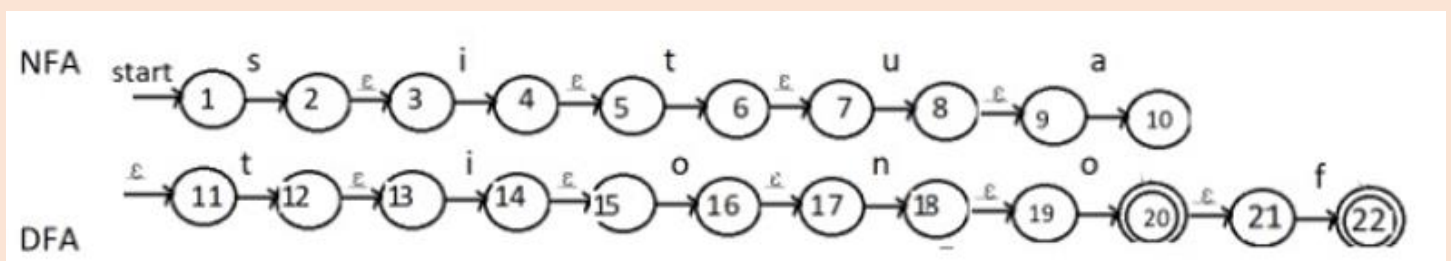


TRANSACTION TABLE :

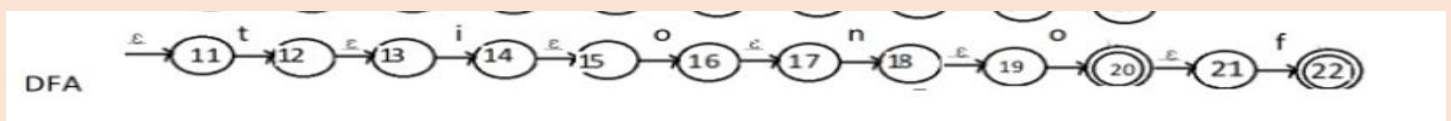
	C	h	e	c	k
1	2'				
2		4'			
4			6'		
6				8'	
8					10'
10					

15-Situation of:

N f a:



D f a:



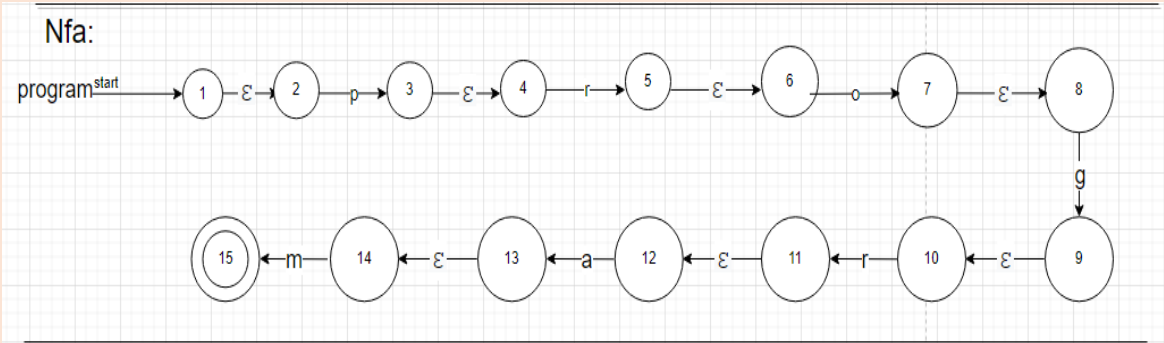
TRANSACTION TABLE :

	s	i	t	u	a	t	i	o	n	o	f
1	2'										
2		4'									
4			6'								
6				8'							
8					10'						
10						12'					
12							14'				
14								16'			
16									18'		
18										20'	
20											22'
22											

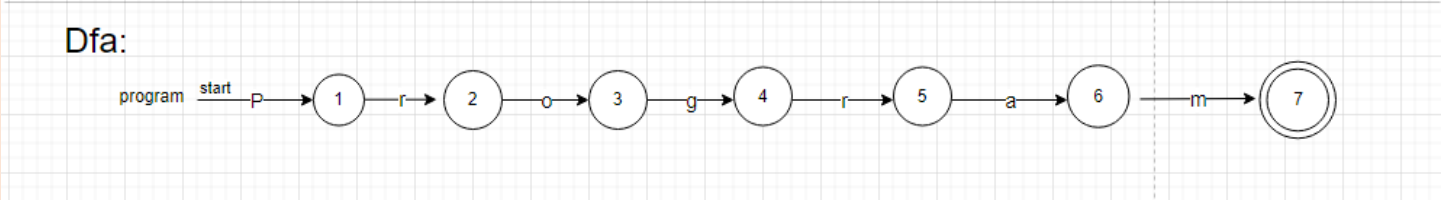
16-Program&End:

regex: program

N f a:



D f a:

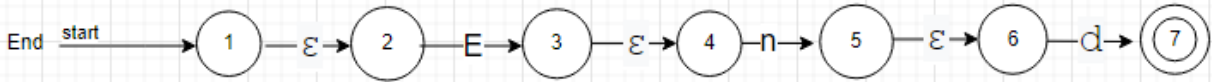


-END:

regex: End

n f a:

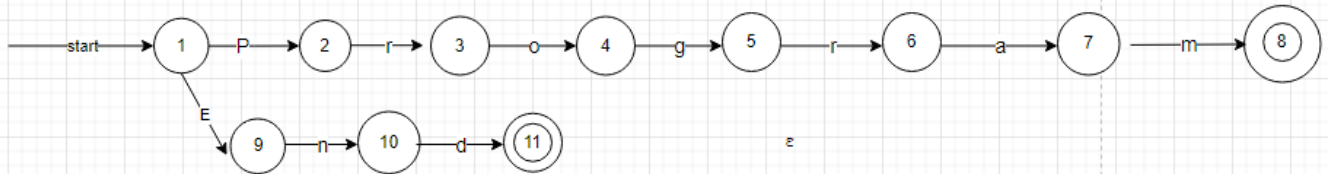
Nfa:



D f a:



DFA of Program&End:

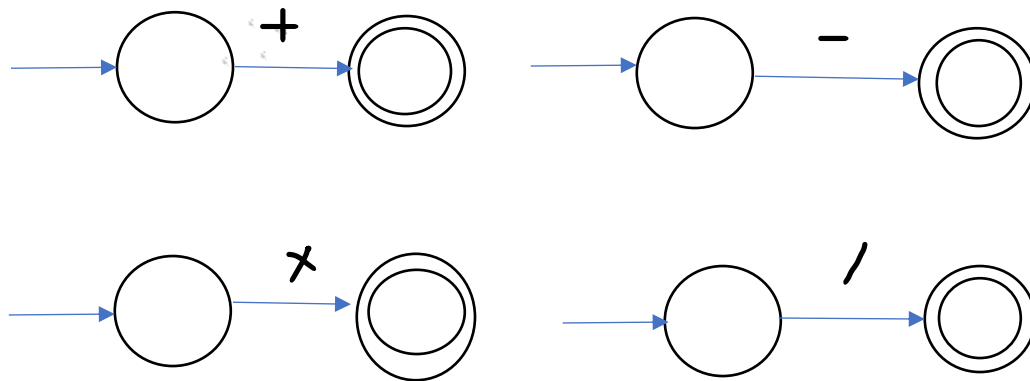


TRANSACTION TABLE :

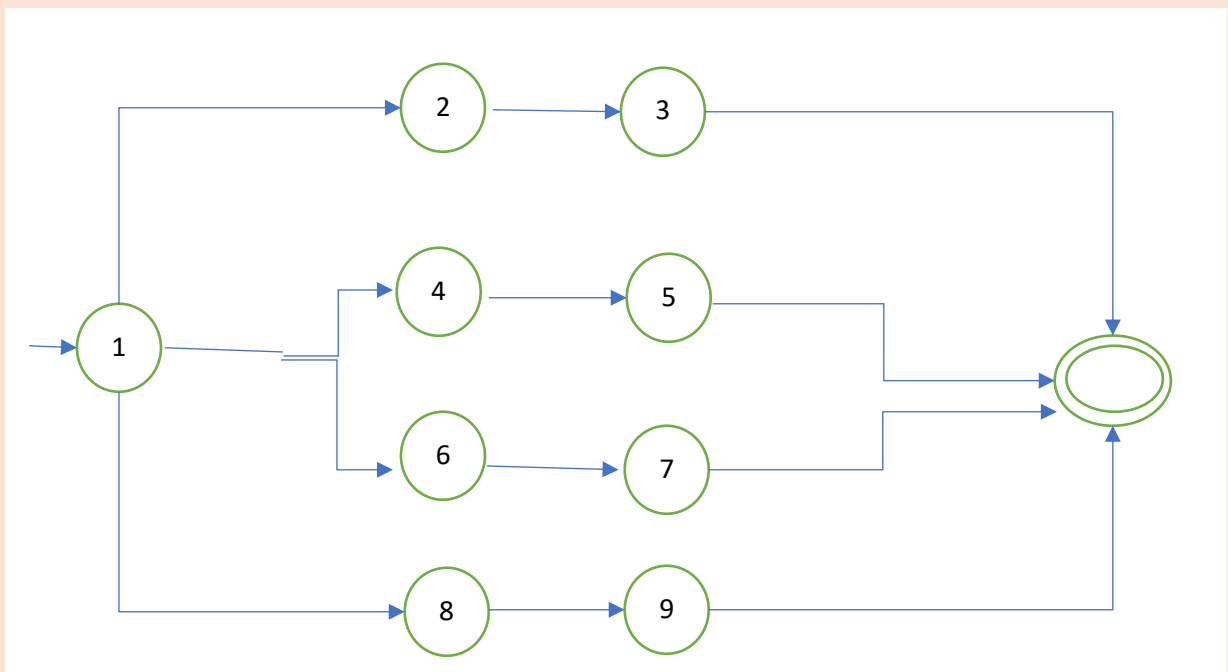
[illegible]

18-Arithmetic Operation:

RE = + | - | * | /



N F A



TRANSACTION TABLE :

1? = {1,2,4,6,8}

2? = {2,3}

3? = {3}

4? = {4,5}

5? = {5}

6? = {6,7}

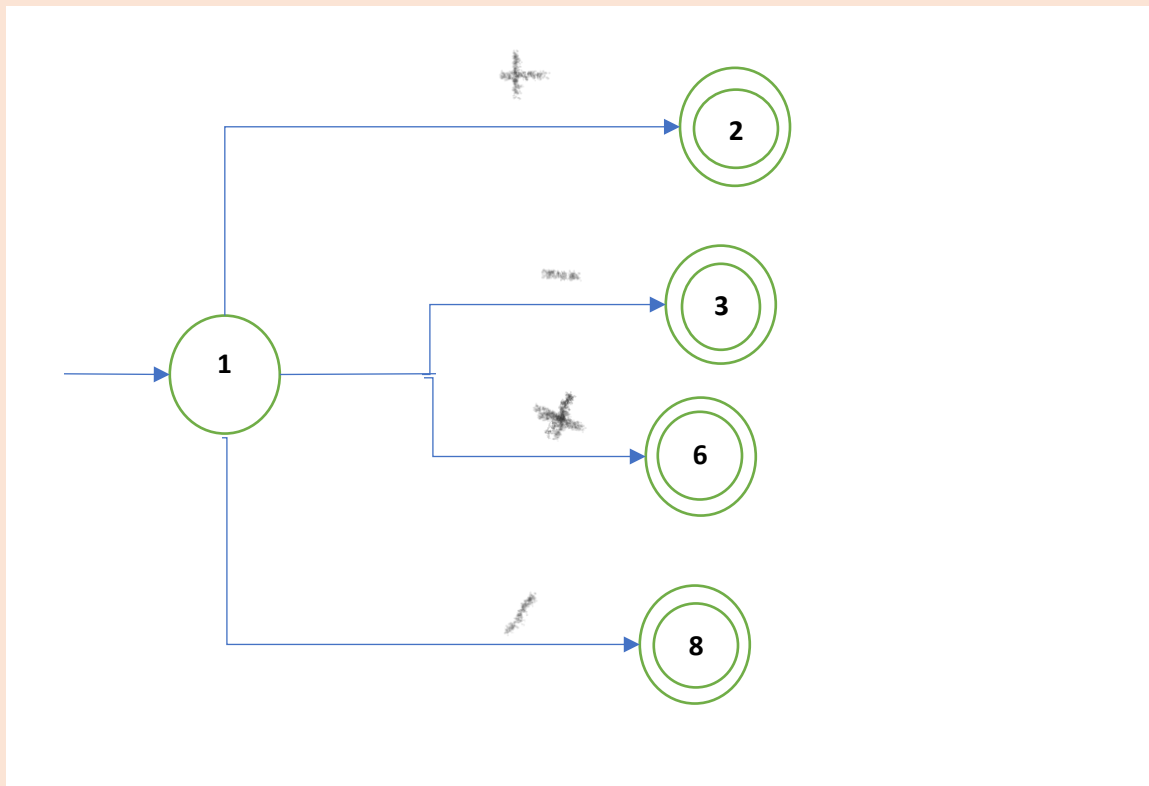
7? = {7}

8? = {8,9}

9? = {9}

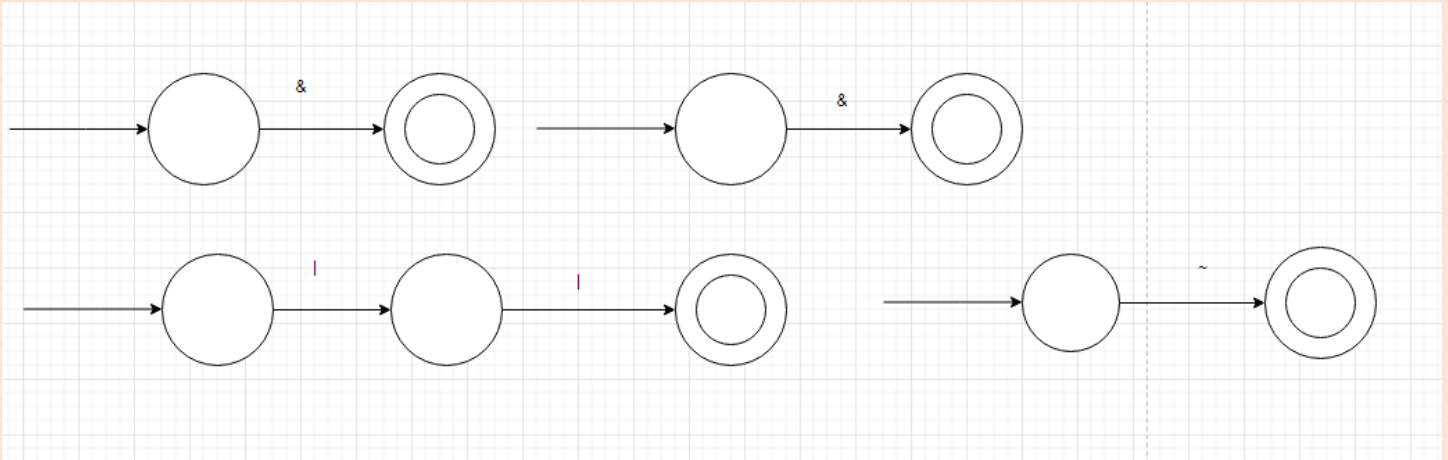
	+	-	*	/
1	2	4	6	8
2				
4				
6				
8				

D F A:

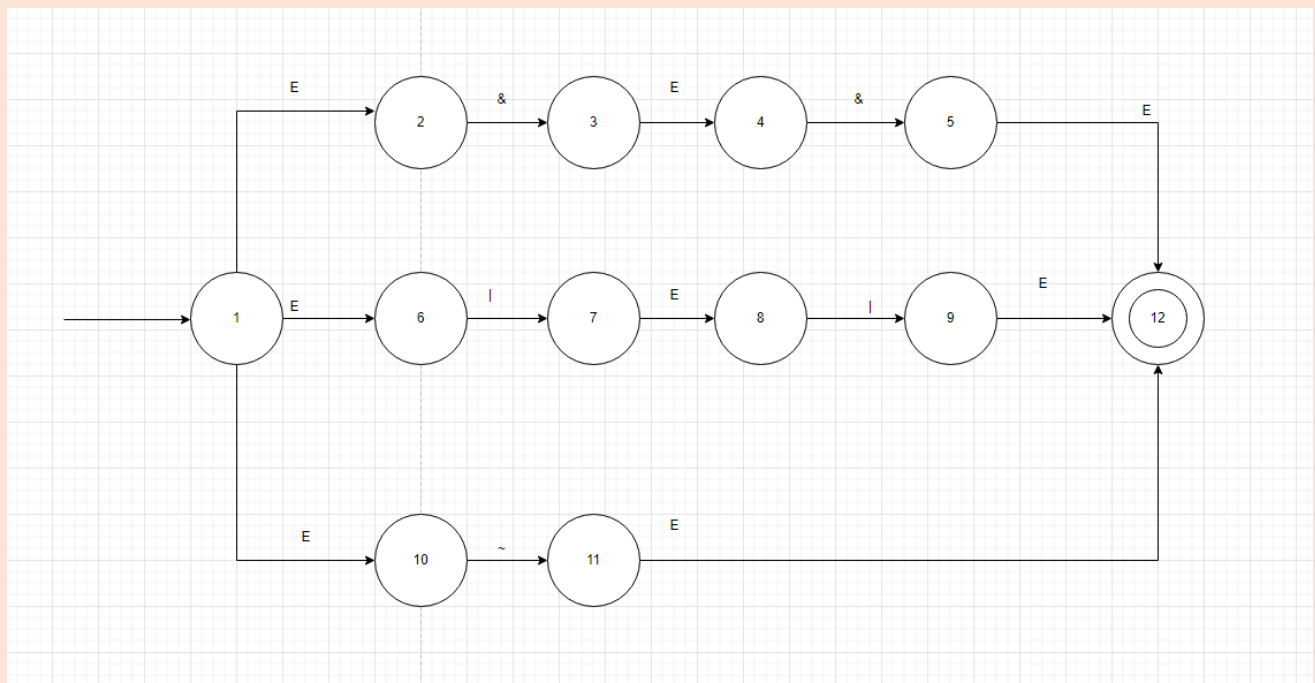


19-Logic operators:

RE = && | || | ~



N F A:



TRANSACTION TABLE :

1? = {1,2,6,10}

2? = {2,3}

3? = {3}

4? = {4,5}

6? = {6,7}

7? = {7}

8? = {8,9}

9? = {9}

5? = {5}

6? = {6,7}

7? = {7}

8? = {8,9}

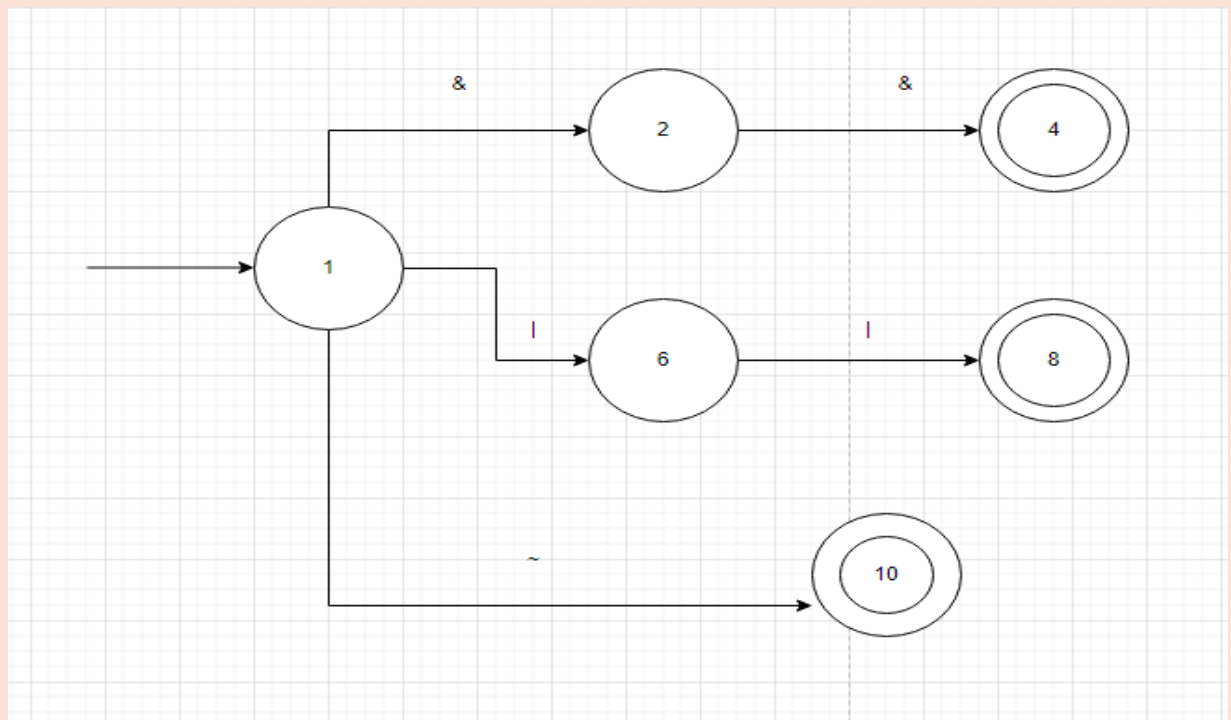
9? = {9}

10? = {10,11}

11? = {11}

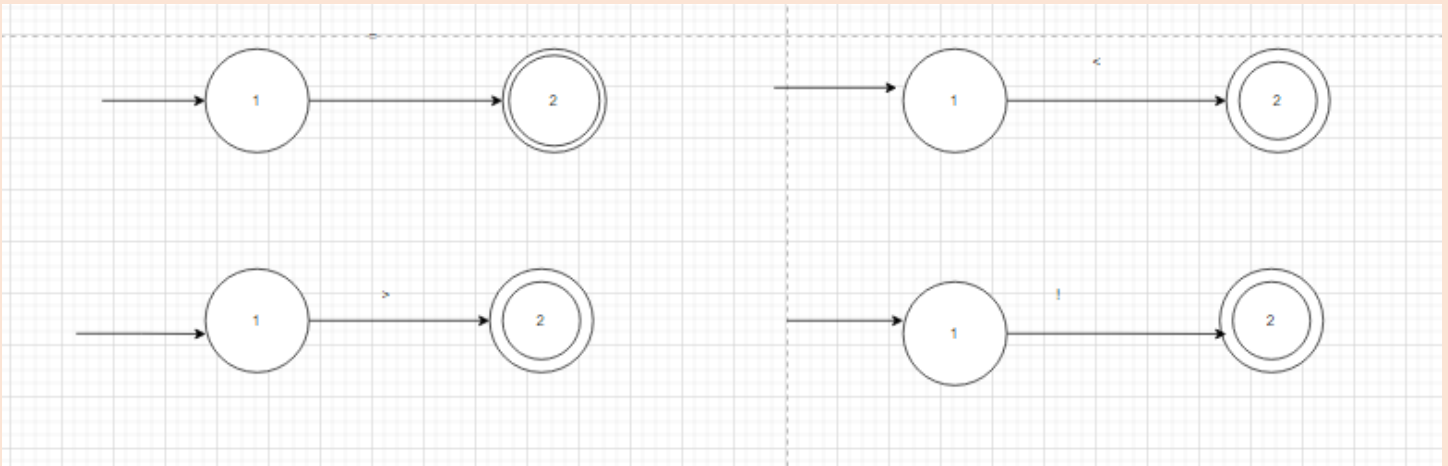
	&	&			~	
1	2		6		10	
2		4				
6				8		

D F A:



20-relational operators:

RE ($=$ | $!$ | $<$ | $>$)

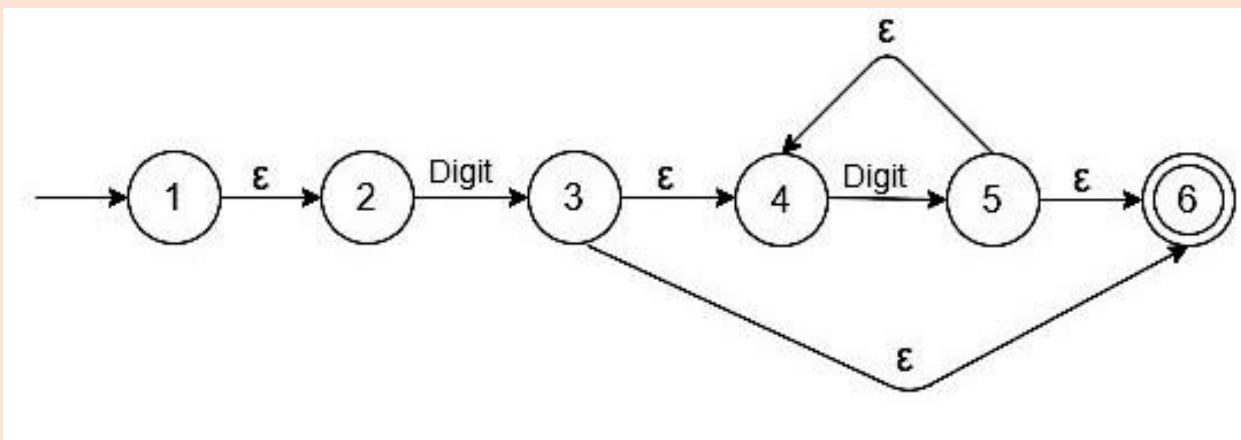


21-Numbers :

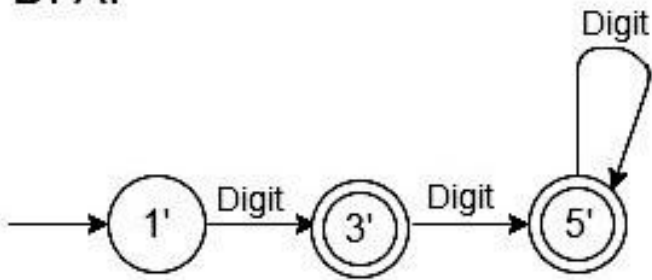
RE = Digit (Digit)*

RE = [0-9]+

NFA:



DFA:

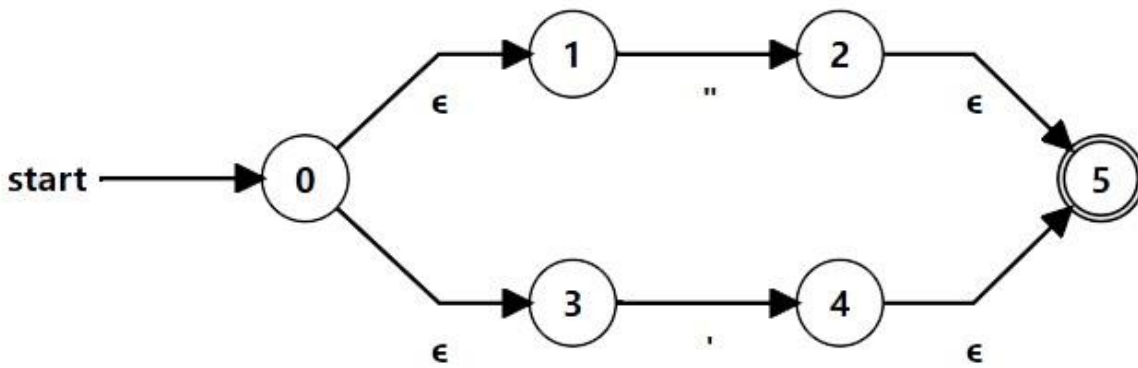


Transition Table	
	Digit
1'	3'
3'	5'
5'	5'

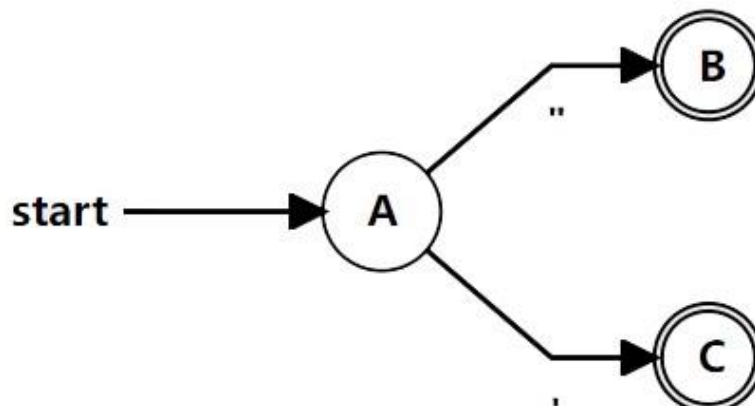
22-Quotation Marks

RE = ("|')

NFA:

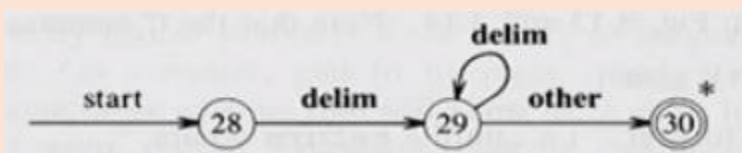
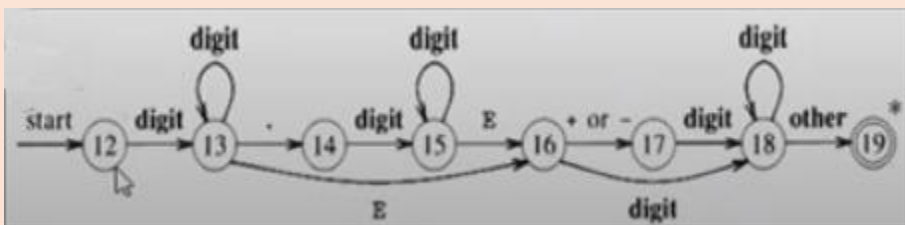
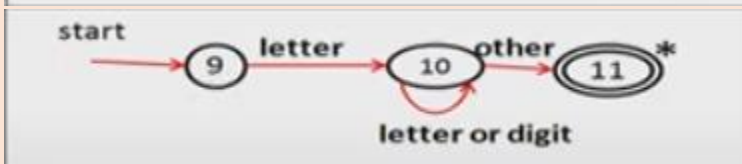
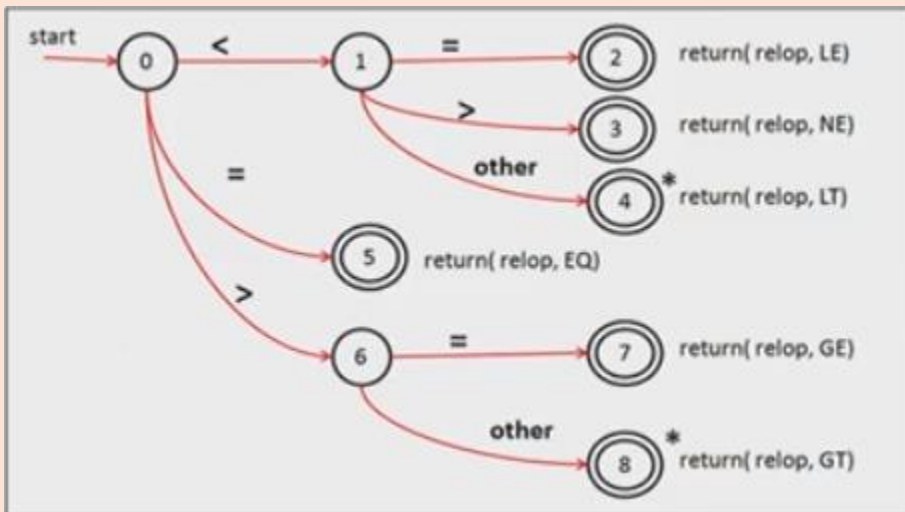


DFA:



-Scanner:

THE TRANSITION TABLE TO IMPLEMENT THE SCANNER:



Parse tree and Abstract syntax tree

1. Program \rightarrow Program ClassDeclaration End.

Program \rightarrow Program`

Program` \rightarrow ClassDeclaration End. Program` | ϵ

First(Program) = First(Program`) = { Category | ϵ }

First(Program`) = { First(ClassDeclaration) , ϵ } = { Category | ϵ }

Follow(Program) = { \$ }

Follow(Program`) = Follow(Program) = { \$ }

2. ClassDeclaration \rightarrow Category ID{ Class_Implementation } | Category ID Derive { Class_Implementation }

ClassDeclaration \rightarrow Category ID ClassDeclaration`

ClassDeclaration` \rightarrow { Class_Implementation } | Derive { Class_Implementation }

First(ClassDeclaration) = { Category }

First(ClassDeclaration`) = { { , Derive } }

Follow(ClassDeclaration) = First(End) = { End }

Follow(ClassDeclaration`) = Follow(ClassDeclaration) = { End }

3. Class_Implementation \rightarrow VarDeclaration Class_Implementation | MethodDeclaration
Class_Implementation | Comment Class_Implementation | using_command
Class_Implementation | Func_Call Class_Implementation | empty

First(Class_Implementation) =

{ Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical , < , - , using , ID , ϵ }

Follow(Class_Implementation) = { }

4. MethodDeclaration \rightarrow Func Decl ; | Func Decl { VarDeclaration Statements }

First(MethodDeclaration) = First(Func Decl) = { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical }

Follow(MethodDeclaration) = {First(Class_Implementation) - ϵ } U Follow(Class_Implementation) = { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical , < , - , using , ID , }

5. Func Decl \rightarrow Type ID (ParameterList)

First(Func Decl) = First(Type) = { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical }

Follow(Func Decl) = { ; , { }

6. Type \rightarrow Ilap | Silap | Clop | Series | Ilapf | Silapf | None | Logical

First(Type) = { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical }

Follow(Type) = { ID }

7. ParameterList \rightarrow empty | None | Non-Empty List

First(ParameterList) = { ϵ , None , Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical }

Follow(ParameterList) = { } }

8. Non-Empty List \rightarrow Type ID | Non-Empty List , Type ID

First(Non-Empty List) = { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical }

Follow(Non-Empty List) = Follow(ParameterList) = { } }

9. VarDeclaration \rightarrow empty | Type ID_List ; VarDeclaration

First(VarDeclaration) = { ϵ , Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical }

Follow(VarDeclaration) = {First(Class_Implementation) - ϵ } U Follow(Class_Implementation) U

{First(Statements) - ϵ } U Follow(MethodDeclaration) U = =

{ Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical , < , - , using , ID , } , Assignment , If , Rotatethen , Continuewhen , terminatethis , read , write , Replywith , = }

10. $ID_List \rightarrow ID \mid ID_List , ID$

$First(ID_List) = \{ ID \}$

$Follow(ID_List) = \{ ; \}$

11. $Statements \rightarrow empty \mid Statement \ Statements$

$First(Statements) = \{ \epsilon , Assignment , If , Rotatewhen , Continuewhen , terminatethis , read , write , Replywith \}$

$Follow(Statements) = \{ \} \}$

12. $Statement \rightarrow Assignment \mid If_Statement \mid Rotatewhen_Statement \mid$
 $Continuewhen_Statement \mid terminatethis_Statement \mid read (ID); \mid write (Expression); \mid$
 $Replywith_Statement$

$First(Statement) = \{ Assignment , If , Rotatewhen , Continuewhen , terminatethis , read , write , Replywith \}$

$Follow(Statement) = \{ First(Statements) - \epsilon \} \cup Follow(Statements)$

$= \{ Assignment , If , Rotatewhen , Continuewhen , terminatethis , read , write , Replywith , \} \}$

13. $Assignment \rightarrow VarDeclaration = Expression;$

$First(Assignment) = \{ First(VarDeclaration) - \epsilon \} \cup First(Expression) =$
 $\{ Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical , = , ID , Number \}$

$Follow(Assignment) = Follow(Statement) =$

$\{ Assignment , If , Rotatewhen , Continuewhen , terminatethis , read , write , Replywith , \} \}$

14. $Func_Call \rightarrow ID (Argument_List) ;$

$First(Func_Call) = \{ ID \}$

$Follow(Func_Call) = \{ First(Class_Implementation) - \epsilon \} \cup Follow(Class_Implementation) =$
 $\{ Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical , < , - , using , ID , \} \}$

15. $\text{Argument_List} \rightarrow \text{empty} \mid \text{NonEmpty_Argument_List}$

$\text{First}(\text{Argument_List}) = \{ \epsilon, \text{ID}, \text{Number} \}$

$\text{Follow}(\text{Argument_List}) = \{ \} \}$

16. $\text{NonEmpty_Argument_List} \rightarrow \text{Expression} \mid \text{NonEmpty_Argument_List}, \text{Expression}$

$\text{First}(\text{NonEmpty_Argument_List}) = \text{First}(\text{Expression}) = \{ \text{ID}, \text{Number} \}$

$\text{Follow}(\text{NonEmpty_Argument_List}) = \text{Follow}(\text{Argument_List}) = \{ \} \}$

17. $\text{Block Statements} \rightarrow \{ \text{statements} \}$

$\text{First}(\text{Block Statements}) = \{ \{ \}$

$\text{Follow}(\text{Block Statements}) = \text{Follow}(\text{If_Statement}) \cup \text{Follow}(\text{Rotate_Statement}) \cup \text{Follow}(\text{Continuewhen_Statement}) =$

$= \{ \text{Assignment}, \text{If}, \text{Rotatewhen}, \text{Continuewhen}, \text{terminatethis}, \text{read}, \text{write}, \text{Replywith}, \} \}$

18. $\text{If_Statement} \rightarrow \text{if}(\text{Condition_Expression}) \text{ Block Statements}$

$\text{First}(\text{If_Statement}) = \{ \text{if} \}$

$\text{Follow}(\text{If_Statement}) = \text{Follow}(\text{Statement}) =$

$\{ \text{Assignment}, \text{If}, \text{Rotatewhen}, \text{Continuewhen}, \text{terminatethis}, \text{read}, \text{write}, \text{Replywith}, \} \}$

19. $\text{Condition_Expression} \rightarrow \text{Condition} \mid \text{Condition Condition_Op Condition}$

$\text{First}(\text{Condition_Expression}) = \text{First}(\text{Condition}) = \text{First}(\text{Expression}) = \{ \text{ID}, \text{Number} \}$

Follow(Condition_Expression) = {) }

20. Condition_Op \rightarrow and | or

First(Condition_Op) = { and , or }

Follow(Condition_Op) = First(Condition) = { ID , Number }

21. Condition \rightarrow Expression Comparison_Op Expression

First(Condition) = First(Expression) = { ID , Number }

Follow(Condition) = Follow(Condition_Expression) \cup Follow(Condition_Op) = {) , ID , Number }

22. Comparison_Op \rightarrow == | != | > | >= | < | <=

First(Comparison_Op) = { = , ! , > , < }

Follow(Comparison_Op) = First(Expression) = { ID , Number }

23. Rotate_Statement \rightarrow Rotate when(Condition_Expression) Block Statements

First(Rotate_Statement) = { Rotate }

Follow(Rotate_Statement) = NA

24. Continuewhen_Statement \rightarrow Continuewhen (expression ; expression ; expression) Block Statements

First(Continuewhen_Statement) = { Continuewhen }

Follow(Continuewhen_Statement) = Follow(Statement) =

{Assignment , If, Rotatewhen , Continuewhen , terminatethis , read , write , Replywith , } }

25. Replywith _Statement → Replywith Expression ; | returnID ;

First(Replywith _Statement) = { Replywith , returnID }

Follow(Replywith _Statement) = Follow(Statement) =

{Assignment , If, Rotatewhen , Continuewhen , terminatethis , read , write , Replywith , } }

26. terminatethis _Statement → terminatethis;

First(terminatethis _Statement) = { terminatethis }

Follow(terminatethis _Statement) = Follow(Statement) =

{Assignment , If, Rotatewhen , Continuewhen , terminatethis , read , write , Replywith , } }

27. Expression → Term | Expression Add_Op Term

First(Expression) = { ID , Number }

Follow(Expression) = { ; ,) , = , ! , > , < }

28. Add_Op → + | -

First(Add_Op) = { + , - }

Follow(Add_Op) = First(Term) = { ID , Number }

29. Term → Factor | Term Mul_Op Factor

First(Term) = { ID , Number }

Follow(Term) = Follow(Expression) = { ; ,) , = , ! , > , < }

30. Mul_Op \rightarrow * | /

First(Mul_Op) = { * , / }

Follow(Mul_Op) = First(Factor) = { ID , Number }

31. Factor \rightarrow ID | Number

First(Factor) = { ID , Number }

Follow(Factor) = Follow(Term) = Follow(Expression) = { ; ,) }

32. Comment \rightarrow < * STR * > | -- STR

First(Comment) = { < , - }

Follow(Comment) = { First(Class_Implementation) - ϵ } \cup Follow(Class_Implementation) =
{ Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical , < , - , using , ID , }

33. using_command \rightarrow using(F_name.txt);

First(using_command) = { using }

Follow(using_command) = { First(Class_Implementation) - ϵ } \cup Follow(Class_Implementation) =
{ Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical , < , - , using , ID , }

34. F_name \rightarrow STR

First(F_name) = { STR }

Follow(F_name) = { .txt }

Text

