CompilerProject Form

**Project Idea: Project #1**

***Description:***A program in Project #1 consists of a sequence of function definitions -.  
Each function consists in turn of variable declarations, type declarations,  
function declarations, and statements. The types in Project #1 are very  
restricted look at table 1.in addition to table 1, single dimensional arrays and  
pointer types are possibly using user defined struct types as in C Language.  
The array index value can only be simple unary expression such as an  
identifier, a constant or another simple array access expression. Project# 1  
Language is case sensitive

……………………………………………………………….

**Team Members NO#:** 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Name** | **Level&**  **Department** | **Section(Day-from-to)** | **Role**  **(Lead/Member**) | **Grade** |
| 201900827 | مصطفى محسن محمد علي | Level3 &CS |  | Leader |  |
| 201900794 | مروان محمد محمود | Level3 &CS |  | Member |  |
| 201900426 | عبدالرحمن عمرو عبدالسلام | Level3 &CS |  | Member |  |
| 201900020 | احمد امير احمد شفيق | Level3 &CS |  | Member |  |
| 201900253 | حازم حافظ محمد | Level3 &CS |  | Member |  |
| 201900151 | اسماء محمود محمد | Level3 &CS |  | Member |  |
| 20180178 | جهاد مصطفى محمد حسن | CS |  | Member |  |

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

# Regular Expression:

|  |  |
| --- | --- |
| Letter | ( [A-Z] | [a-z] ) |
| Digit | [ 0-9 ] |
| Digits | Digit + |
| Sign | + | - |
| Condition | If—Else |
| Integer | Iow |
| SInteger | SIow |
| Character | Chlo |
| String | Chain |
| Float | Iowf |
| SFloat | SIowf |
| Void | Worthless |
| Loop | (Loop | Iterate) when |
| Return | Turnback |
| Break | Stop |
| Struct | Loli |
| ArithmeticOperation | (+| -| \*| /) |
| Logic  Operators | (&& | || | ~) |
| relational operators | (== | < | > | != | <= | >=) |
| Assignment operator | = |
| Access Operator | -> |
| Braces | ({ | } | [ | ]) |
| Constant | Digits |
| Quotation Mark | , |
| Inclusion | Include |



if

# NFA:-



0



1



2



i



f

# DFA:-



0



1



2



i



f

# Else

## NFA:-



0



1



4



E



2



3



s



l



e

## DFA:-



0



1



4



E



2



3



s



l



e

# Iow

## NFA:-



1



2



3



o



w



0



I

## DFA:-



1



2



3



o



w



0



I

# SIow

## NFA:-



0



1



4



S



2



3



o



I



w

## DFA:-



0



1



4



S



2



3



o



I



w

# Chlo

## NFA:-



0



1



4



C



2



3



l



h



o

## DFA:-



0



1



4



C



2



3



l



h



o

# Chain

## NFA:-



0



1



5



C



2



3



a



h



i



4



n

## DFA:-



0



1



5



C



2



3



a



h



i



4



n

# Iowf

## NFA:-



0



1



3



I



2



3



w



o



f

## DFA:-



0



1



4



I



2



3



w



o



f

# SIowf

## NFA:-



0



1



5



S



2



3



o



I



w



4



f

## DFA:-



0



1



5



S



2



3



o



I



w



4



f

Worthless

NFA:

-

DFA:

-



0



1



W



2



3



r



o



t



4



h



5



6



9



l



7



8



s



e



s



0



1



W



2



3



r



o



t



4



h



5



6



9



l



7



8



s



e



s

Loopwhen

NFA:

-

DFA:

-



0



1



L



2



3



o



o



p



4



w



5



6



8



h



7



n



e



0



1



L



2



3



o



o



p



4



w



5



6



8



h



7



n



e

# Turnback

## NFA:-



0



1



T



2



3



r



u



n



4



b



5



6



8



a



7



k



c

## DFA:-



0



1



t



2



3



r



u



n



4



b



5



6



8



a



7



k



c

# Stop

## NFA:-



0



1



4



S



2



3



o



t



p

## DFA:-



0



1



4



S



2



3



o



t



p

# Loli

## NFA:-



0



1



4



L



2



3



l



o



i

## DFA:-



0



1



4



L



2



3



l



o



i

**Arithmetic Operation**

## NFA:-



0



1



**ε**



2



**ε**



+



3



**ε**



4



**ε**



-



5



**ε**



6



**ε**



\*



7



**ε**



8



**ε**



/



9

DFA:

-



0



**+**



**-**



**\***



**/**



1



2



3



4



Logic operators

## NFA:-



0



1



**ε**



2



&



4



**ε**



5



|



7



**ε**



8



~



9



6



|



3



&



**ε**



**ε**



**ε**

## DFA:-



0



2



4



5



1



3



&



&



|



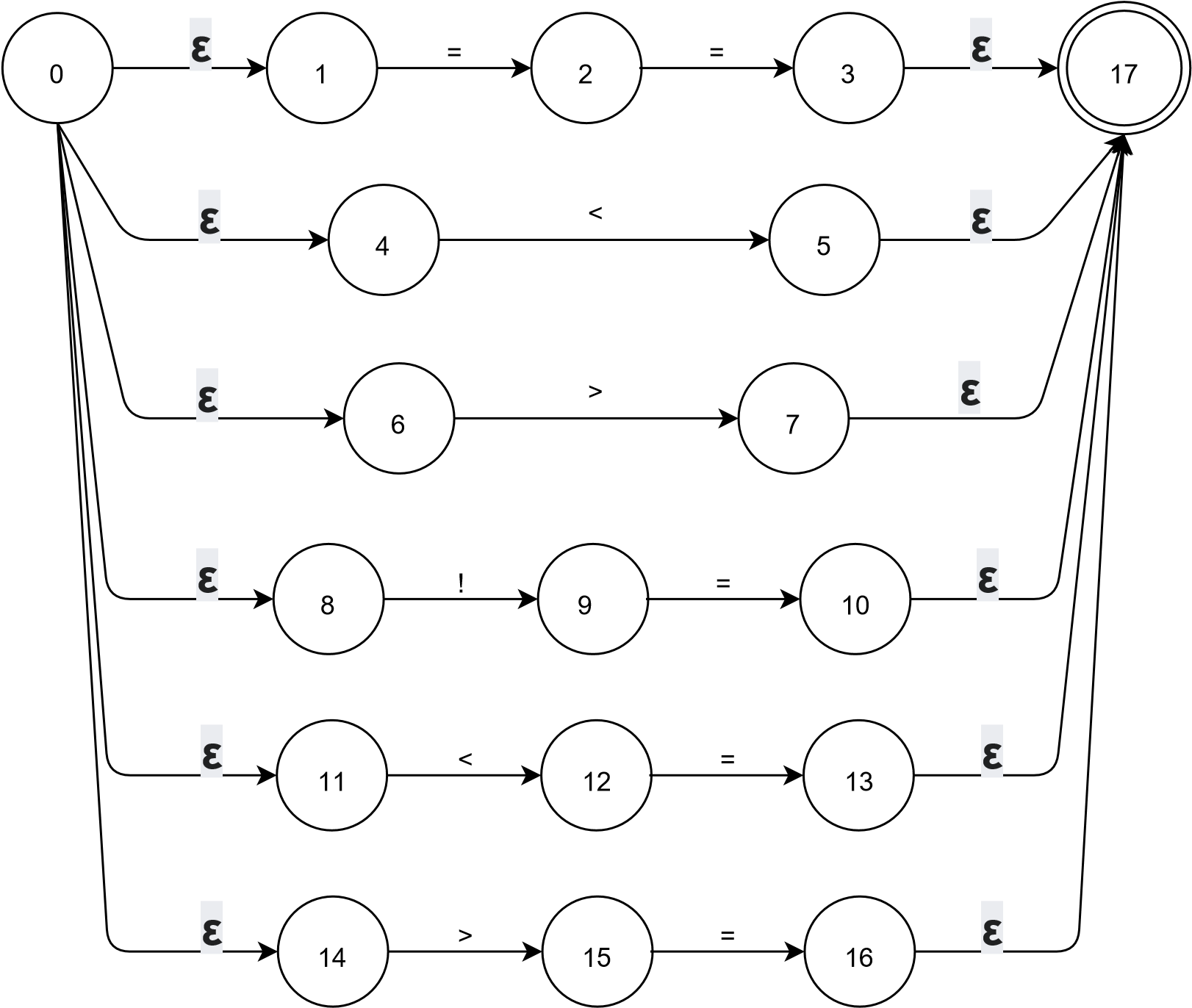
|



~

relational operators

## NFA:-



## DFA:-



0



2



4



1



=



=



<



>



6



5



!



=



8



7



<



=



10



9



>



=



3

Assignment operator

## NFA:-



1



0



=

## DFA:-



1



0



=

Access Operator

## NFA:-



2



0



1



-



>

## DFA:-



2



0



1



-



>



Braces

## NFA:-



0



1



**ε**



2



**ε**



{



3



**ε**



4



**ε**



}



5



**ε**



6



**ε**



]



7



**ε**



8



**ε**



[



9

## DFA:-



0



**{**



**}**



**]**



**[**



1



2



3



4

Constant



NFA:

-



0



1



3



**ε**



2



**ε**



[0

-

9]



[0

-

9]



DFA:

-



1



0



[0

-

9]



[0

-

9]

Quotation Mark



0



1



**ε**



2



**ε**



'



4



**ε**



5



**ε**



"



3



NFA:

-



DFA:

-



0



'



"



2



1

Include



## NFA:-



0



1



I



2



3



c



n



l



4



u



5



6



7



d



e

## DFA:-



1



2



7



**ε**



3



6



**ε**



]



A

-

Z



[



**ε**



0

NFA:

-



4



**ε**



5



**ε**



[



a

-

z



]



**ε**



0



1



I



2



3



c



n



l



4



u



5



6



7



d



e

|  |  |
| --- | --- |
| String |  |



DFA:

-



1



0



2



[



A

-

Z



]



a

-

z



]



[



[



A

-

Z



]



[



]



a

-

z



[



a

-

z



]



[



A

-

Z



]

Single Line Comment

## NFA:-



1



2



3



$



$



0



$

## DFA:-



1



2



3



$



$



0



$

Multi Line Comment

## NFA:-



2



3



4



$



/



1



$



0



/

## DFA:-



2



3



4



$



/



1



$



0



/

First Function () :

First(program)= {*Iow, SIow, Chlo, Chain, Iowf, SIowf ,Worthless, /\*, ///, include}*

First(declaration-list) ={ *Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless }*

First(declaration-list’) ={ *Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , ϵ }*

First(declaration) = { *Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , /\*, /// }*

First(var-declaration) = { *Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless }*

First(type-specifier) = { *Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless}*

First(fun-declaration)={ *Iow, SIow, Chlo, Chain, Iowf, SIowf, Worthless , /\*, /// }*

First(params) = { *Iow, SIow, Chlo, Chain, Iowf, SIowf, Worthless , ϵ }*

First(params-list) = { *Iow, SIow, Chlo, Chain, Iowf, SIowf, Worthless }*

First(params-list’) = { , , *ϵ }*

First(param) = { *Iow, SIow, Chlo, Chain, Iowf, SIowf, Worthless }*

First(compound-stmt) = *{* ***{*** *}*

First(compound-stmt’) = { */\*, /// , Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , ϵ }*

First(local-declarations) = { *Iow , SIow , Chlo , Chain , Iowf , SIowf ,Worthless , ϵ }*

First(statement-list) = { *ID , ( , + , - , INT\_NUM , FLOAT\_NUM , ; , { , if , Loopwhen ,*

*Iteratewhen , Turnback , Stop , ϵ }*

First(statement) = { *ID , ( , + , - , INT\_NUM , FLOAT\_NUM , ; , { , if , Loopwhen ,*

*Iteratewhen , Turnback , Stop }*

First(expression-stmt) = { *ID , ( , + , - , INT\_NUM , FLOAT\_NUM , ; }*

First(selection-stmt) = { *if }*

First(selection-stmt’) = { *else , ϵ }*

First(iteration-stmt) = { *Loopwhen , Iteratewhen }*

First(loop-statement) = { *Loopwhen }*  
First(iterate-statement) = { *Iteratewhen }*

First(jump-stmt) = { *Turnback , Stop }*

First(expression) = { *ID , ( , + , - , INT\_NUM , FLOAT\_NUM }*

First(expression) = { *=* , *ϵ }*

First(id-assign) = { *ID }*

First(Simple-expression) = { *( ,*  *ID , + , - , INT\_NUM , FLOAT\_NUM }*

First(simple-expression-cont) = { *<=, <, >, >=, ==, !=, &&, ||, ε }*

First(relop) = { *<= , < , > , >= , == , != , && , || }*

First(additive-expression) = { *( ,*  *ID , + , - , INT\_NUM , FLOAT\_NUM }*

First(additive-expression) = { + , - , *ε }*

First(addop) = { *+ , - }*

First(term) = { *( ,*  *ID , + , - , INT\_NUM , FLOAT\_NUM }*

First(term’) = { \* , / , *ε }*

First(mulop) = { \* , / *}*

First(factor) = { *( ,* *ID , + , - , INT\_NUM , FLOAT\_NUM }*

First(call) = { *ID }*

First(args) = { *ID , ( , + , - , INT\_NUM , FLOAT\_NUM , ε }*

First(arg-list) = { *ID , ( , + , - , INT\_NUM , FLOAT\_NUM }*

First(arg-list) = { , , *ε }*

First(num) = { *+ , - , INT\_NUM , FLOAT\_NUM }*

First(unsigned-num) = { *INT\_NUM , FLOAT\_NUM }*

First(unsigned-num) = { *+ , - }*

First(pos-num) = { *+ }*  
First(neg-num) = { *- }*

First(value) = { *INT\_NUM , FLOAT\_NUM }*

First(comment) = { */\* , /// }*

First(include\_command) = { *include }*

First(f\_name) = { *STR }*

Follow Function () :

Follow (program) = {$}

Follow (declaration-list) = Follow (program) = {$}

Follow (declaration-list’) = Follow (declaration-list) ={$}

Follow (declaration) ={ first(declaration-list’)- ε} U follow ( declaration-list)= { Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , $}

Follow (var-declaration) = follow(declaration) U follow (*local-declarations*) = { Iow ,

SIow , Chlo , Chain , Iowf , SIowf , Worthless , $, *ID , ( , + , - , INT\_NUM , FLOAT\_NUM , ; ,*

*{ , if , Loopwhen , Iteratewhen , Turnback , Stop* }

Follow (type-specifier) ={ first (params) - ε} U first (compound-stmt ) U follow (fun-declaration) = {ID , ( , *Iow, SIow, Chlo, Chain, Iowf, SIowf, Worthless ,)* ,}}

Follow ( fun-declaration) = follow(declaration) = { Iow , SIow , Chlo , Chain , Iowf ,

SIowf , Worthless , $}

Follow (params)= first (compound-stmt ) U follow (fun-declaration) = {) , {, Iow , SIow ,

Chlo , Chain , Iowf , SIowf , Worthless , $,}}

Follow (param-list) = Follow (params)= {) , {, Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , $}

Follow (param-list’) = Follow (param-list) = {) , {, Iow , SIow , Chlo , Chain , Iowf ,

SIowf , Worthless , $}

Follow (param) = {first (param-list’) - ε} U follow (param-list) U follow (param-list’) = { , , )

, {, Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , $}

Follow (compound-stmt) = Follow ( fun-declaration) U follow (statement) = { Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , $, }}

Follow (compound-stmt’) = Follow (compound-stmt) = { Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , $,}}

Follow (local-declarations) = {first(statement-list) - ε} U first(var-declaration) U follow (compound-stmt’) = { *ID , ( , + , - , INT\_NUM , FLOAT\_NUM , ; , { , if , Loopwhen ,*

*Iteratewhen , Turnback , Stop* Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , $}

Follow (statement-list) = follow(}) {}}

Follow (statement) = Follow (statement-list) U first( selection-stmt) U follow

(selection-stmt) U first (selection-stmt’)= {},if }

Follow (expression-stmt) = Follow (statement-list) = {}}

Follow (selection-stmt) = Follow (statement) = {},if}

Follow (selection-stmt’) = Follow (selection-stmt) = {},if}

Follow (iteration-stmt) = Follow (statement) = {},if}

Follow (Loop-statement) = Follow (iteration-stmt) = {},if}

Follow (Iterate -statement) = Follow (iteration-stmt) = {},if}

Follow (jump-stmt) = Follow (statement) = {},if}

Follow (expression) = follow(;) U follow ())U first(statement) U{ first(selection-stmt’) - ε }

U follow(Loop-statement) U follow(expression’) U {first(arg-list’)- ε }U follow (arg-list’)= {;,),ID , ( , + , - , INT\_NUM , FLOAT\_NUM , { , if , Loopwhen , Iteratewhen , Turnback , Stop

, else, },if ,, }

Follow (expression’) = Follow (expression) = {;,),ID , ( , + , - , INT\_NUM , FLOAT\_NUM ,

{ , if , Loopwhen , Iteratewhen , Turnback , Stop , else, },if ,, }

Follow (id-assign) = {first (expression’) - ε} U Follow (expression) = { = , ;}

Follow (simple-expression) = Follow (expression) = {;,),ID , ( , + , - , INT\_NUM ,

FLOAT\_NUM , { , if , Loopwhen , Iteratewhen , Turnback , Stop , else, },if ,, }

Follow (simple-expression’) = Follow (simple-expression) = {;}

Follow (relop) = first (additive-expression) U follow (simple-expression’) = { ( , ID , + , - , INT\_NUM , FLOAT\_NUM , ; }

Follow (additive-expression) = { first(simple-expression’) - ε } U follow (simple-expression)

= { <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (additive-expression’) = Follow (additive-expression) = { <=, <, >, >=, ==, !=, &&, ||,

; }

Follow (addop) = first(term) U {first (additive-expression’) - ε} U follow

(additive-expression’) = { ( , ID , + , - , INT\_NUM , FLOAT\_NUM , + , - , <=, <, >, >=, ==,

!=, &&, ||, ; }

Follow (term) = {first(additive-expression’) - ε } U follow (additive-expression) U follow (additive-expression’) = { + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (term’) = Follow (term) = { + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (mulop) = first ( factor) U {first (term’) – ε } U follow (term’) = { ( , ID , + , - ,

INT\_NUM , FLOAT\_NUM , \* , / , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow ( factor ) = { first (term’) – ε } U follow (term) U follow ( term’) = { \* , / , + , - , <=,

<, >, >=, ==, !=, &&, ||, ; }

Follow (call) = follow (factor) = { \* , / , + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (args) = { ) }

Follow ( arg-list ) = follow ( args ) = { ) }

Follow (arg-list’)= follow (arg-list)={ ) }

Follow (num) = follow ( factor ) = { \* , / , + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow ( Unsigned num ) = follow ( num) = { \* , / , + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (Signed num) = follow ( num) = { \* , / , + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (pos-num) = follow (Signed num) = { \* , / , + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (neg-num) = follow (Signed num) = { \* , / , + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (value) = follow ( Unsigned num ) U follow ( pos-num ) U follow (neg-num) = { \* ,

/ , + , - , <=, <, >, >=, ==, !=, &&, ||, ; }

Follow (comment) = follow (program) U first ( type-specifier ) U ID U follow   
(fun-declaration) U {first(local-declarations) - ε } U { first (statement-list) - ε } U follow (compound-stmt’) = {$ , Iow , SIow , Chlo , Chain , Iowf , SIowf , Worthless , ID , ( , + , - , INT\_NUM , FLOAT\_NUM , ; , { , if , Loopwhen , Iteratewhen , Turnback , Stop }

Follow ( include\_command ) = follow ( program ) = { $ }

Follow (F\_name) = Follow ( include\_command ) = { $ }