1. Change to the folder which include fp.l and run the below command

> **java -jar jflex-1.6.1.jar fp.l**

**Compilation Sample:**

Constructing NFA : 217 states in NFA

Converting NFA to DFA :

..........................................................................................................

110 states before minimization, 91 states in minimized DFA

Old file "Lexer.java" saved as "Lexer.java~"

Writing code to "Lexer.java"

2. Compile the yacc code with the below command.

> **java -jar javacup11b.jar -interface < FP.y**

Compilation sample:

------- CUP v0.11b 20150226 (SVN rev 63) Parser Generation Summary -------

0 errors and 7 warnings

35 terminals, 23 non-terminals, and 45 productions declared,

producing 78 unique parse states.

4 terminals declared but not used.

0 non-terminals declared but not used.

0 productions never reduced.

0 conflicts detected (0 expected).

Code written to "parser.java", and "sym.java".

---------------------------------------------------- (CUP v0.11b 20150226 (SVN rev 63))

3. Then you will get Lexer.java file. Compile it with the below command.

>**javac Lexer.java**

4. Then we can use sample.fp to test the generated lexer.

>**java Lexer sample.fp**

**Test case1:**

============

Input:

{Program sample

{Function factory VAL

{if {< VAL 0}

then {= retVal -1}

else {= retVal 1}

{while { < VAL 0} do

{= retVal {\* retVal VAL}}

{= VAL {- VAL 1}}

}

}

return retVal

}

{print {facto 999}}

}

**Lexer output with no syntax error:**

----------------------------------

code:4 {

line:1 col:2 --{--BRAC\_O--

code:19 Program

line:1 col:3 --Program--PROGRAM--

code:11, lexem :sample

line:1 col:11 --sample--IDENTIFIER--

code:4 {

line:2 col:2 --{--BRAC\_O--

code:20 Function

line:2 col:3 --Function--FUNCTION--

code:11, lexem :factory

line:2 col:12 --factory--IDENTIFIER--

code:11, lexem :VAL

line:2 col:20 --VAL--IDENTIFIER--

code:4 {

line:3 col:2 --{--BRAC\_O--

code:22 if

line:3 col:3 --if--IF--

code:4 {

line:3 col:6 --{--BRAC\_O--

code:18 <

line:3 col:7 --<--LESS--

code:11, lexem :VAL

line:3 col:9 --VAL--IDENTIFIER--

code:33, lexem : 0

line:3 col:12 -- 0--INTEGER--

code:5 }

line:3 col:14 --}--BRAC\_C--

code:23 then

line:4 col:2 --then--THEN--

code:4 {

line:4 col:7 --{--BRAC\_O--

code:16 =

line:4 col:8 --=--EQUAL--

code:11, lexem :retVal

line:4 col:10 --retVal--IDENTIFIER--

code:33, lexem :-1

line:4 col:17 ---1--INTEGER--

code:5 }

line:4 col:19 --}--BRAC\_C--

code:24 else

line:5 col:2 --else--ELSE--

code:4 {

line:5 col:7 --{--BRAC\_O--

code:16 =

line:5 col:8 --=--EQUAL--

code:11, lexem :retVal

line:5 col:10 --retVal--IDENTIFIER--

code:33, lexem : 1

line:5 col:16 -- 1--INTEGER--

code:5 }

line:5 col:18 --}--BRAC\_C--

code:4 {

line:6 col:2 --{--BRAC\_O--

code:25 while

line:6 col:3 --while--WHILE--

code:4 {

line:6 col:9 --{--BRAC\_O--

code:18 <

line:6 col:11 --<--LESS--

code:11, lexem :VAL

line:6 col:13 --VAL--IDENTIFIER--

code:33, lexem : 0

line:6 col:16 -- 0--INTEGER--

code:5 }

line:6 col:18 --}--BRAC\_C--

code:26 do

line:6 col:20 --do--DO--

code:4 {

line:7 col:2 --{--BRAC\_O--

code:16 =

line:7 col:3 --=--EQUAL--

code:11, lexem :retVal

line:7 col:5 --retVal--IDENTIFIER--

code:4 {

line:7 col:12 --{--BRAC\_O--

code:9 \*

line:7 col:13 --\*--MULTIPLY--

code:11, lexem :retVal

line:7 col:15 --retVal--IDENTIFIER--

code:11, lexem :VAL

line:7 col:22 --VAL--IDENTIFIER--

code:5 }

line:7 col:25 --}--BRAC\_C--

code:5 }

line:7 col:26 --}--BRAC\_C--

code:4 {

line:8 col:2 --{--BRAC\_O--

code:16 =

line:8 col:3 --=--EQUAL--

code:11, lexem :VAL

line:8 col:5 --VAL--IDENTIFIER--

code:4 {

line:8 col:9 --{--BRAC\_O--

code:8 -

line:8 col:10 -----MINUS--

code:11, lexem :VAL

line:8 col:12 --VAL--IDENTIFIER--

code:33, lexem : 1

line:8 col:15 -- 1--INTEGER--

code:5 }

line:8 col:17 --}--BRAC\_C--

code:5 }

line:8 col:18 --}--BRAC\_C--

code:5 }

line:9 col:2 --}--BRAC\_C--

code:5 }

line:10 col:2 --}--BRAC\_C--

code:21 return

line:11 col:2 --return--RETURN--

code:11, lexem :retVal

line:11 col:9 --retVal--IDENTIFIER--

code:5 }

line:12 col:2 --}--BRAC\_C--

code:4 {

line:13 col:2 --{--BRAC\_O--

code:29 print

line:13 col:3 --print--PRINT--

code:4 {

line:13 col:9 --{--BRAC\_O--

code:11, lexem :facto

line:13 col:10 --facto--IDENTIFIER--

code:33, lexem : 999

line:13 col:15 -- 999--INTEGER--

code:5 }

line:13 col:19 --}--BRAC\_C--

code:5 }

line:13 col:20 --}--BRAC\_C--

code:5 }

line:14 col:2 --}--BRAC\_C--

line:14 col:4 ----EOF--

5. Compile Main.java with below command:

>**javac Main.java**

6. Test the sample input with the below command:

>**java Main sample.fp**

**Code Output:**

-----------

Mytest

code:4 {

code:19 Program

code:11, lexem :sample

code:4 {

code:20 Function

code:11, lexem :factory

code:11, lexem :VAL

code:4 {

code:22 if

code:4 {

code:18 <

code:11, lexem :VAL

code:33, lexem : 0

code:5 }

code:23 then

code:4 {

code:16 =

code:11, lexem :retVal

code:33, lexem :-1

code:5 }

code:24 else

code:4 {

code:16 =

code:11, lexem :retVal

code:33, lexem : 1

code:5 }

code:4 {

code:25 while

code:4 {

code:18 <

code:11, lexem :VAL

code:33, lexem : 0

code:5 }

code:26 do

code:4 {

code:16 =

code:11, lexem :retVal

code:4 {

code:9 \*

code:11, lexem :retVal

code:11, lexem :VAL

code:5 }

code:5 }

code:4 {

code:16 =

code:11, lexem :VAL

code:4 {

code:8 -

code:11, lexem :VAL

code:33, lexem : 1

code:5 }

code:5 }

code:5 }

code:5 }

code:21 return

code:11, lexem :retVal

code:5 }

code:4 {

code:29 print

code:4 {

code:11, lexem :facto

code:33, lexem : 999

code:5 }

code:5 }

code:5 }

BLOCK0:: --> factory:2 --> <:4 --> else:3 --> <:8 --> facto:14

BLOCK1:: --> then:3 --> =:5 --> =:6 --> do:7 --> =:9 --> =:11

BLOCK2:: --> sample:1 --> VAL:2 --> if:3 --> VAL:4 --> retVal:5 --> retVal:6 --> while:7 --> VAL:8 --> retVal:9 --> \*:10 --> retVal:10 --> VAL:10 --> VAL:11 --> VAL:12 --> return:2 --> retVal:2 --> print:13

BLOCK3:: --> Program:1 --> Function:2

BLOCK4::0

**Test Case 2 (Syntax Error):**

============

**Sample input(Removed less than symbol from line /\*{if { VAL 0}\*/):**

-------------

{Program sample

{Function factory VAL

**{if { VAL 0}**

then {= retVal -1}

else {= retVal 1}

{while { < VAL 0} do

{= retVal {\* retVal VAL}}

{= VAL {- VAL 1}}

}

}

return retVal

}

{print {facto 999}}

}

**Sample Output:**

==============

>**java Main sample.fp**

Mytest

code:4 {

code:19 Program

code:11, lexem :sample

code:4 {

code:20 Function

code:11, lexem :factory

code:11, lexem :VAL

code:4 {

code:22 if

code:4 {

code:11, lexem :VAL

**Error: Syntax error @ Symbol: IDENTIFIER (unknown:3/8(-1) - unknown:3/10(-1))**

**instead expected token classes are [GREATEREQ, NOTEQ, GREATER, LESS]**

**Fatal: Couldn't repair and continue parse**

**Can't recover from previous error(s), giving up. @ Symbol: IDENTIFIER (unknown:3/8(-1) - unknown:3/10(-1))**