

College of Engineering & Technology

***Computer Science Department***

# COMP439 Final Project S23/24 Due midnight : 12-6-2024

# =================================================================

***Given the grammar which almost represents a subset of MODULA-2 programming language :***

module-decl 🡪 module-heading declarations block name .

module-heading 🡪 **module** name ;

block 🡪 **begin** stmt-list **end**

declarations 🡪 const-decl var-decl procedure-decl

const-decl 🡪 **const** const-list | λ

const-list 🡪 name = value ; const-list | λ

var-decl 🡪 **var** var-list | λ

var-list 🡪 var-item ; var-list | λ

var-item 🡪 name-list : data-type

name-list 🡪 name more-names

more-names 🡪 , name-list | λ

data-type 🡪 **integer | real | char**

procedure-decl 🡪 procedure-heading declarations block name ; procedure-decl | λ

procedure-heading 🡪 **procedure** name ;

stmt-list 🡪 statement ; stmt-list | λ

statement 🡪 ass-stmt | read-stmt | write-stmt | if-stmt | while-stmt

| loop-stmt | exit-stmt | call-stmt | block | λ

ass-stmt 🡪 name := exp

exp 🡪 term exp-prime

exp-prime 🡪 add-oper term exp-prime | λ

term 🡪 factor term-prime

term-prime 🡪 mul-oper factor term-prime | λ

factor 🡪 “(“ arith-exp “)” | name-value

add-oper 🡪 + | -

mul-oper 🡪 \* | / | **mod** | **div**

read-stmt 🡪**readint** “(“ name-list “)” | **readreal** “(“ name-list “)”

| **readchar** “(“ name-list “)” | **readln**

write-stmt 🡪**writeint** “(“ write-list “)” | **writereal** “(“ write-list “)”

**writechar** “(“ write-list “)” | **writeln**

write-list 🡪 write-item more-write-value

more-write-value 🡪 , write-list | λ

write-item 🡪 name | value

if-stmt 🡪 **if** condition **then** stmt-list else-part **end**

else-part 🡪 **else** stmt-list | λ

while-stmt 🡪 **while** condition **do** stmt-list **end**

loop-stmt 🡪 **loop** stmt-list **until** condition

exit-stmt 🡪 **exit**

call-stmt 🡪 **call** name (\* This is a procedure name \*)

condition 🡪 name-value relational-oper name-value

relational-oper 🡪 = | |= | < | <= | > | >=

name-value 🡪 name | value

value 🡪 integer-value | real-value

**Keep in mind:**

**name** is generated by the regular expression: letter ( letter | digit )\*

**integer-value** is generated by the regular expression: digit ( digit )\*

**real-value** is generated by the regular expression: digit ( digit )\*. digit ( digit )\*

The tokens in **bold** are reserved words or standard identifiers (library functions or procedures).

**Write an LL(1) predictive parser for the above grammar, that is, using LL(1) parsing table.**

**Note:**

1. You are allowed **only** to use JAVA programming language, any other language will not accepted and you get 0 mark.

2.  you can **only** use Java **fx** or Java **swing**  for UI if necessary.

3. You are allowed only to use built in Java packages, DONT use any open source libraries.

4. Make sure your code well commented.

5. Submitted files with .java extension, any other extension will not be accepted.

6. Put your files in one folder and name it with your (name - id).

7. Make sure not to share your work with others and using any AI tools, you will get 0 mark.

8. You should work **individually only**, any signs of cheating whatsoever will be penalized severely.

9. No programs will be accepted after the deadline for any reason.

10. Your program will be tested with a random files of mine.

11. Submit your project by replying to the message “**439-Project-S23**” on Ritaj web page.