Birzeit University logo

College of Engineering & Technology

***Computer Science Department***

# COMP439 Final Project Fall 23/24 Due midnight : 19-01-2024

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

***Given the grammar defined by the following set of production rules which represent a subset of MODULA-2 programming language :***

module-decl 🡪 module-heading declarations procedure-decl block name .

module-heading 🡪 **module** name ;

block 🡪 **begin** stmt-list **end**

declarations 🡪 const-decl var-decl

const-decl 🡪 **const** const-list | λ

const-list 🡪 ( name **=** value **;** )\*

var-decl 🡪 **var** var-list | λ

var-list 🡪 ( var-item **;** )\*

var-item 🡪 name-list  **:** data-type

name-list 🡪 name ( **,** name )\*

data-type 🡪 **integer | real | char**

procedure-decl 🡪 procedure-heading declarations block name ;

procedure-heading 🡪 **procedure** name ;

stmt-list 🡪 statement ( ; statement )\*

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

| while-stmt | repeat-stmt | exit-stmt | call-stmt | λ

ass-stmt 🡪 name := exp

exp 🡪 term ( add-oper term )\*

term 🡪 factor ( mul-oper factor )\*

factor 🡪 “(“ 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 ( **,** write-item )\*

write-item 🡪 name | value

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

elseif-part 🡪 ( **elseif** condition **then** stmt-list )\*

else-part 🡪 **else** stmt-list | λ

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

repeat-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

name-value 🡪 name | value

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

name 🡪 letter ( letter | digit )\*

value 🡪 integer-value | real-value

integer-value 🡪 digit ( digit )\*

real-value 🡪 digit ( digit )\*. digit ( digit )\*

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

Write an a **recursive descent** parser for the above grammar.

\* Use only **Java** as a host language.

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

\* Your program will be tested with my text files (programs), either it works correctly or doesn’t.

\* No programs will be accepted after the deadline for any reason whatsoever.

\* In the ERROR function, report the error clearly and precisely showing the **line** and **token** where

the Error occurs and exit the program (panic mode) error handling.

\* Submit only the source code by replying to the message “**439-Project-F23**” on Ritaj web page.