Faculty of Computer & Information Sciences

Ain Shams University

Subject: Compiler Theory

Year: 3rd level undergraduate

Academic year: 1st term 2021-2022

**Milestone (2)-Task (1)-CFG Rules**

Extra regular expressions:

Digit:=[0-9]

Letter:=[A-Za-z]

Statements:= (Condition\_Statement|Comment\_Statement|Function\_Call|Read\_Statement| Return\_Statement?|

If\_Statement|Else\_Statement|Else\_If\_Statement|

Repeat\_Statement| Write\_Statement|

Assignment\_Statement|Declaration\_Statement)

Statements -> Condition\_Statement | Comment\_Statement

| Function\_Call | Read\_Statement | **A** |If\_Statement

| Else\_Statement | Else\_If\_Statement

| Repeat\_Statement | Write\_Statement

| Assignment\_Statement |Declaration\_Statement

**A** -> Return\_Statement | ԑ

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1)Number:=^Digit+(\.Digit+)?

2)String:=^”(.\*)”$

3)Reserved\_Keywords:= (int | float | string | read | write | repeat | until | if | elseif | else | then | return | endl | end)

4)Comment\_Statement:=^( \/ \\*) (.\*)(\\*\/)$

5)Identifier:=Letter(Letter|Digit)\*

6)Function\_Call:=Identifier\((Identifier(,Identifier)\*)?\)$

Function\_Call -> Identifier (ArgList)

ArgList -> Identifier Arguments | ԑ

Arguments ->, ArgList | ԑ

7)Term:=(Number|Identifier|Function\_Call)

Term -> Number|Identifier|Function\_Call

8)Arithmetic\_Operator:= (+ | - | \* | / )

9) Equation := Term|\(Term (Arithmetic\_operator Term)+\)

Equation -> Term | (EQ1) EQ2

EQ1 -> Term EQ2

EQ2 -> Arithmetic\_operator Term EQ2| ԑ

10) Expression := (String|Term|Equation)

Expression -> String|Term|Equation

11) Assignment\_Statement := Identifier \:\= Expression

Assignment\_Statement -> Identifier := Expression

12) Datatype := (int|float|string)

Datatype -> int|float|string

13) Declaration\_Statement :=

^Datatype (identifier | Assignment\_Statement)

(, identifier |, Assignment Statement)\*;$

Declaration\_Statement -> Datatype A ;

A -> (identifier | Assignment\_Statement) B

B -> , A | ԑ

14)Write\_Statement:=^write Expression(endl)?;$

Write\_Statement -> write Expression Endline;

Endline -> endl | ԑ

15)Read\_Statement:=^read (Identifier) ;$

Read\_Statement -> read Identifier ;

16)Return\_Statement := ^return (Expression);$

Return\_Statement -> return Expression;

17)Condition\_Operator := (< | > | = | <>)

18)Condition := ^(Identifier) (Condition\_Operator) (Term)

Condition -> Identifier Condition\_Operator Term

19)Boolean\_Operator := ( && | \|\| )

20)Condition\_Statement := ^(Condition) [(Boolean\_Operator)(Condition)]\*

Condition\_Statement -> Condition A

A -> Boolean\_Operator Condition A | ԑ

21)If\_Statement := ^if (Condition\_Statement) then (Statements) (Else\_If\_Statement )\* (Else\_Statement)? end$

If\_Statement -> if (Condition\_Statement) Then Statements A B end

A -> Else\_If\_Statement A | ԑ

B -> Else\_Statement | ԑ

22)Else\_If\_Statement := ^elseif (Condition\_Statement) then (Statements) (Else\_If\_Statement )\* (Else\_Statement)? end$

Else\_If\_Statement -> elseif (Condition\_Statement) Then Statements A B end

A -> Else\_If\_Statement A | ԑ

B -> Else\_Statement | ԑ

23)Else\_Statement := ^else (Statements) end$

Else\_Statement -> else Statements end

24)Repeat\_Statements := ^repeat (Statements) until (Condition\_Statement)

Repeat\_Statements -> repeat Statements until Condition\_Statement

25) FunctionName:= Identifier

FunctionName -> Identifier

26) Parameter:= (Datatype)(Identifier)

Parameter -> Datatype Identifier

27) Function\_Declaration:= (Datatype)(Identifier)

\( ( Parameter(\,Parameter)\*)? \)

Function\_Declaration -> Datatype Identifier Function\_Declaration\_second

Function\_Declaration\_second -> (Function\_Declaration\_third) | ԑ

Function\_Declaration\_third -> Parameter Function\_Declaration\_fourth

Function\_Declaration\_fourth ->, Function\_Declaration\_third | ԑ

28) Function\_Body:= \{ [Statements]+ (Return\_Statement)\}

Function\_Body -> { States } Return\_Statement

States -> Statements States\_second

States\_second -> Statements States\_second | ԑ

29)Function\_Statement:=(Function\_Declaration) (Function\_Body)

Function\_Statement -> Function\_Declaration Function\_Body

30)Main\_Function := (Datatype) main \( \) (Function\_Body)

Main\_Function -> Datatype main ( ) Function\_Body

31)Program:= (Function\_Statement)\* (Main\_Function)

Program -> Function\_state Main\_Function

Function\_state -> Function\_Statement Function\_state | ԑ