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 | If\_Statement

| Else\_Statement | Else\_If\_Statement

| Repeat\_Statement | Write\_Statement

| Assignment\_Statement |Declaration\_Statement

**|** Return\_Statement

States -> Statements States\_repeatition

States\_repeatition -> Statements States\_repeatition | 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:=Function\_Name\((Expression(,Expression)\*)?\)$

Function\_Call -> Function\_Name (ArgList)

ArgList -> Expression Arguments | ԑ

Arguments ->, ArgList | ԑ

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

Term -> Number|Identifier|Function\_Call

Left Factoring: Term -> Number|Identifier D

D-> ԑ | (ArgList)

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

9) Equation := Term (Arithmatic\_operator Term)\* |

((Term Arithmetic\_operator )\*

\(Term (Arithmetic\_operator Term)+\)

(Arithmetic\_operator Term) \*)+

Equation -> Term | (Equation\_A) Equation\_B

Equation\_A -> Term Equation\_B

Equation\_B -> Arithmetic\_operator Term Equation\_B| ԑ

Equation -> Equation Addop Term | Term

Addop -> + | -

Term -> Term Mulop Factor | Factor

Mulop -> \* | /

Factor -> (Equation) | number

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

Expression -> String|Term|Equation

Left Factoring: Expression -> String|Term D

D-> ԑ | 3abbath

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 (identifier | Assignment\_Statement) Declare\_A ;

Declare\_A -> ,(identifier | Assignment\_Statement) Declare\_A| ԑ

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

Write\_Statement -> write (Expression | endl);

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 Condition\_State

Condition\_State -> Boolean\_Operator Condition Condition\_State | ԑ

21)If\_Statement := ^if (Condition\_Statement) then

[Statements]+ (Else\_If\_Statement | Else\_Statement| end)

If\_Statement -> if (Condition\_Statement) Then States (Else\_If\_Statement | Else\_Statement| end)

22)Else\_If\_Statement := ^elseif (Condition\_Statement) then [Statements]+ (Else\_If\_Statement | Else\_Statement | end)

Else\_If\_Statement -> elseif (Condition\_Statement) Then States (Else\_If\_Statement | Else\_Statement| end)

23)Else\_Statement := ^else [Statements]+ end$

Else\_Statement -> else States end

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

Repeat\_Statements -> repeat States 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 (Parameters\_list)

Parameter\_list -> Parameter Parameter\_repeatition | ԑ

Parameter\_repeatition ->, Parameter Parameter\_repeatition | ԑ

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

Function\_Body -> { States Return\_Statement }

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 -> User\_Function Main\_Function

User\_Function -> Function\_Statement User\_Function | ԑ

9) Equation := Term (Arithmatic\_operator Term)\* |

((Term Arithmetic\_operator )\*

\(Term (Arithmetic\_operator Term)+\)

(Arithmetic\_operator Term) \*)+

Equation -> Term | (Equation\_A) Equation\_B

Equation\_A -> Term Equation\_B

Equation\_B -> Arithmetic\_operator Term Equation\_B| ԑ

Arithmatic\_op -> + | - | \* | /

Equation -> Term Equation\_a

Equation\_a -> Arithmatic\_op Term Equation\_a | ԑ

Term -> Factor Term\_a

Term\_a -> Arithmatic\_op Factor Term\_a | ԑ

Factor -> (Equation) | number