COMPILER'S DESIGN

Milestone 1

Group members:

- Abdallah Hesham – 179674
- Mirna Victor – 190860
- Mostafa Mahfouz – 182004
- Hossam Hassan - 180871

“TINY” Language Regular Expressions:

Number:
Digit:= [0-9]
Num_Un_Signed := (Digit)+
Num_Signed := (+|-)? Num_Un_Signed
Num_float := Num_Signed (\.Num_Un_Signed)?

String:
Letter = [a-z]|[A|Z]
Str:= ^\".*\"$

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

Comment_Statement:
L_Comment := ^/*.**/$

Identifiers:
identifier := Letter(Letter|Digit)*

Term:
Term := (Num_float | identifier | Fun_call)

Function_Call:
Fun_call := identifier \(((identifier)(, identifier)*)? \)

Reserved_Keywords:
R_Keywords := int|float|string|read|write|repeat|until|if|elseif|else|then|return|end

Equation:
E_unit = (Term+ Arth_op)*(Term+)$
Equ = E_unit | (Term Arth_op)* \(E_unit \)(Arth_op Term)*

Arithmatic_Operator:
Arth_op = (+ | - | * | /)

Expression:
Exp := Term|Str|Equ

Assignment_Statement:
Ass_st:= (identifier := Exp)

Declaration_Statement:
Dec_st := ^Datatype identifier (,identifier|,Ass_st)*;$

Write_Statement:
Write_st:= ^ write (EXp|\n) ;$

Read_Statement:
Read_st:= ^ read identifier ;$

Return_Statement:

Return_st:= ^ return Exp ;$

Condition Operator:
Con_op:= (<|>|=|<>)

Condition:
Con:= (identifier Con_op term)

Boolean Operator:
Boolean_Op:= (&& | ||)

Condition Statement:
Condition (Boolean_Operator Condition)*

Set_of_Statements:
Set_of_Statements := (Assignment_Statement | Declaration_Statement | Write_Statement | Read_Statement | (Return_Statement)? | Function_Call)

If Statement:
If_Statement := "if" Condition_Statement "then" Set_of_Statements (Else_If_Statement | Else_Statement | end)

Else_If_Statement:
Else_If_Statement := "elseif" Condition_Statement then Set_of_Statements (Else_If_Statement | Else_Statement | "end")

Else_Statement:
Else_Statement := "elseif" Condition_Statement "then" Set_of_Statements (Else_If_Statement | Else_Statement | "end")

Repeat_Statement:
Repeat_Statement := "repeat" Set_of_Statements "until" Condition_Statement

FunctionName:
FunctionName := Identifier

Parameter:
Parameter := Datatype Identifier

Function_Body:
Function_Body := { Set_of_Statements (Return_Statement) }

Function_Declaration:
Function_Declaration :=
Datatype FunctionName \((Parameter(,Parameter)*)? \)

Program:
Program:= (Function_Statment)* Main_Function

Function_Statement:
Function_Statement:= Function_Declaration Function_Body

Main_Function:
Main_Function := Datatype "main" \(\ ) Function_Body