MODIFICATION OF KPL’S LEXICAL RULES

**I/Rules for comments**

* Single-line comment : Starts with //, end at end of line.
* Multi-line Comments : between /\* and \*/

**II/Rules for identifiers**

* A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.

1. The first letter of an identifier should be either a letter or an underscore.
2. You cannot use keywords as identifiers.
3. 3.There is no rule on how long an identifier can be. However, only 15 characters are significant
4. 4.Identifiers are case-insensitive"

* A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.

1. The first letter of an identifier should be either a letter or an underscore.
2. You cannot use keywords as identifiers.
3. There is no rule on how long an identifier can be. However, only 15 characters are significant
4. Identifiers are case-sensitive

* A valid identifier can have letters (both uppercase and lowercase letters) and digits. The underscore is considered as a special letter.

1. You cannot use keywords as identifiers.
2. There is no rule on how long an identifier can be. However, only 15 characters are significant
3. Identifiers are case-insensitive"

* A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.

1. The first letter of an identifier should be either a letter or an underscore.
2. You cannot use keywords as identifiers.
3. Length of an identì?er is not over than 15.
4. Identifiers are case-insensitive"

* A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.

1. The first letter of an identifier should be either a letter or an underscore.
2. You cannot use keywords as identifiers.
3. Length of an identì?er is not over than 15.
4. Identifiers are case-sensitive

* A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.

1. The first letter of an identifier should be a letter
2. You cannot use keywords as identifiers.
3. Length of an identì?er is not over than 15.
4. Identifiers are case-insensitive"

* A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.

1. The first letter of an identifier should be a letter
2. You cannot use keywords as identifiers.
3. Length of an identì?er is not over than 15.
4. Identifiers are case-sensitive

* A valid identifier can have letters (both uppercase and lowercase letters), digits, the dollar sign ($) and the underscore (\_)

1. The first letter of an identifier should be either a letter or an underscore.
2. You cannot use keywords as identifiers.
3. Length of an identì?er is not over than 15.
4. Identifiers are case-sensitive

* A valid identifier can have letters (both uppercase and lowercase letters), digits, the dollar sign ($) and the underscore (\_)

1. The first letter of an identifier should be either a letter or an underscore.
2. You cannot use keywords as identifiers.
3. Length of an identì?er is not over than 15.
4. Identifiers are case-insensitive

**III/Rules for Keywords**

* All keywords must be written in lowercase
* All keywords must be written in uppercase
* Keywords are case-insensitive

**IV/ Rules for symbols**

* Assign operator =, relop: equal to ==
* Relop: not equal to <>
* Relop: not equal to => 2 choices <> or ><
* Relop: not equal to ><
* Indexing expression enclosed by [ and ] instead of (. and .)

**V/ Rules for char/string constants**

* String constants are allowed in KPL. A string constant can contain a sequence of printable symbols, length not exceeding 255, enclosed in single quotes. A single quote is written as ''''
* Use an addition character \ for 2 constants single quote and backslash, that is ‘\’’ for a single quote and ‘\\’for backslash.

**VI/ Syntax Errors Handling**

* Modify file example2.kpl so that it causes the error “Invalid statement”
* Modify file example2.kpl so that it causes the error “Invalid constant!"
* Modify file example2.kpl so that it causes the error “Invalid term”
* Modify file example2.kpl so that it causes the error “Invalid arguments”
* Modify file example1.kpl so that it causes the error "Invalid type!"
* Modify file example2.kpl so that it causes the error "Invalid parameter!"
* Modify file example2.kpl so that it causes the error ""Invalid basic type!"""
* Modify file example2.kpl so that it causes the error "Invalid factor!"
* Modify file example2.kpl so that it causes the error "Invalid comparator!"
* Modify file example2.kpl so that it causes the error "Missing …."

**VII Syntax Rules**

1/Rewrite function compileStatement (and function compileCallSt if necessary) in case the procedure calls do not contain the keyword CALL, i.e. the syntax rule for procedure calls as follows:

CallSt ::= ProcedureIdent Arguments

2/

Given the new set of rules for Arguments:

Arguments ::= SB\_LPAR Arguments1 SB\_RPAR

Arguments1 ::= Expression Arguments2

Arguments1 ::= ε

Arguments2::= SB\_COMMA Expression Arguments2

Arguments2::= ε

Modify relevant functions in your parser to match the new rules

2/ Suppose syntax of declarations in KPL follows the following new rules

Block ::= Declarations Block2

Declarations::= ConstDecls Declarations|TypeDecls Declarations VarDecls Declarations|SubDecls Declarations|ε

Constdecls ::= KW\_CONST ConstDecl ConstDecls2

ConstDecls2::= ConstDecl ConstDecls2

ConstDecls2::= ε

TypeDecls ::= KW\_TYPE TypeDecl TypeDecls2

TypeDecls2 ::= TypeDecl TypeDecls2

TypeDecls ::= ε

VarDecls ::= KW\_VAR VarDecl VarDeclrs2

VarDecls2 ::= VarDecl VarDecls

VarDecls2 ::= ε

SubDecls ::= FunDecl SubDecls|ProcDecls SubDecls|ε

FunDecl ::= KW\_FUNCTION TK\_IDENT Params SB\_COLON BasicType SB\_SEMICOLON Block SB\_SEMICOLON

ProcDecl ::= KW\_PROCEDURE TK\_IDENT Params SB\_SEMICOLON Block SB\_SEMICOLON

Block2 ::= KW\_BEGIN Statements KW\_END

Modify relevant functions in your parser to match the new rules

3/Given the new set of rules for Block4, Block5 and Block6

Block4 ::= FunDecls Block5

Block4 ::= Block5

Block5 ::= ProcDecls Block6

Block5 ::= Block6

FunDecls::= FunDecl FunDecls

FunDecls::= ε

ProcDecls::= ProcDecl ProcDecls

ProcDecls ::= ε

Block6 ::= KW\_BEGIN Statements KW\_END

Modify relevant functions in your parser to match the new rules

4/ Write the function compileParams1 and rewrite the related function to satisfy the following syntax rules <Params>::= SB\_LPAR <Params1> SB\_RPAR

<Params1>::= <Param> <Params2>

<Params1>::= ε

<Params2>::= SB\_SEMICOLON <Param> <Params2>

<Params2>::= ε

5/ Write the function compileVariables2 and rewrite the related function to satisfy the following syntax rules <VarDecl>::= TK\_IDENT <Variables2> SB\_COLON <Type> SB\_SEMICOLON

<Variables2>::=TK\_IDENT <Variables2>

<Variables2>::= ε