Francisco José de Caldas District University

Science Computing III

Professor: Carlos Andrés Sierra Virgüez
Student: Gabriela Martínez Eslava - 20202020117

WORKSHOP N° 2

1. For each one of next cases define a regular expressions as used in a compiler based on the Python re library.

a) Identifier: A regular expression to match valid identifiers (variable names, function names, etc.)

```
IDENTIFIER= re. compile (r'\b [a-zA-Z_][a-zA-Z0-9_]* \b'
```

b) Integer literal: A regular expression to match literals

```
INTEGER = re. compile (r'\b [-]? \d+\b)
```

c) Floating Point Literal: A regular expression to match floating point literals.

```
FLOATING = re.compile (r'\b [-]?\d+\.\ d+\b')
```

d) String literal: A regular expression to match string literals enclosed in double quotes.

```
STRING= re.compile (r'\b\" [^\"] * \" )
```

e) Single line comments: A regular expression to match single line comments starting with "//".

```
COMMENT = re. compile (r'\b //. * \b')
```

f) multi-line comments: A regular expression to match Single-line comments enclosed in '/* */'.

```
COMMENT = re. compile (r' \b /\*. * ? \* /\b')
```

g) whitespace: A regular expression to match whitespace characters (spaces, tabs, newlines).

```
WHITESPACE = re. compile (r'\s +')
```

2.a. X= 5+ 3 * 2 ;

```
S → Program
  → Statement List
  → Statement  StatementList
  → Assignment
  → Identifier "=" Expression ";"
  → X = Term Expression' ";"
  → X = Factor Term' "+" Term Expression'
  → X = 5 + Factor Term'
  → X = 5 + 3 * 2
```

2.b if (x>0) {y = X - 1 ;} else {y =0 }

```
S → Program
  → StatementList
  → Statement StatementList
  → ifStatement
  → "if" "(" Expression")" "{" StatementList "}" Elsepart
  → "if" "(" Term Expression' ")" "{" Statement Statement list "}" Else part
  → if ( Factor Term') { Assingment } else {Statement List}
  → if (identifier > Factor Term') { identifier = Expression ; } else {Statement list}
  → if (X > Number) {y = Term Expression'; } else {Statement Statement List}
  → if (x >0) {y = Factor Term' - Term Expression ; } else { Assingment}
  → if (x >0) {y = identifier - Factor Term' ; } else {Identifier = Expression;}
  → if (X >0) {y = x - Number; } else { y = Term Expression'; }
  → if (X>0) {y = X- 1; } else {y = Factor term'; }
  → if (x >0) { y = x - 1 } else { y = number }
  → if (x >0) { y = x - 1 } else { y = 0; }
```

2.c. while (x<10) { x=x+1; }

```
S → Program
  → Statement list
  → Statement  Statement list
  → whilestatement
  →"while "(" Expression")" "{" Statement List "}"
  → while ( Term Expression' ) { Statement  Statement list }
  → while ( Factor Term' < Term Expression ) { Assingment }
  → while ( Identifier < Factor Term' Term Expression { Identifier "=" Expression ";" }
  → while ( x < Number ) { x = Term Expression' ; }
  → while ( x <10) { x = Factor Term' "+" Term expression`; }
  → while ( x <10) { x = Identifier + factor Term'; }
  → while (x <10) { x = x + number; }
  → while (x<10) { x = x+10; }
```

2.d. return (a + b) * c ;

```
S → Program
  → Statement list
  → Statement  Statementlist
  → Return statement
  → return  Expression
  → return  Term  expression`;
  → return  Factor  Term`;
  → return (Expression) *  Factor Term';
  → return (Term  expression') * Identifier;
  → return (factor Term` + Term  Expression ) * c;
  → return (Identifier + factor Term`) * c ;
  → return ( a + Identifier) * c;
  → return (a + b) * c;
```