WorkShop No. 2 — Compilers

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Members:

Andrés Felipe Salazar Malagón – 20202020043

Teacher:

Carlos Andrés Sierra Virguez

Universidad Distrital Francisco José de Caldas

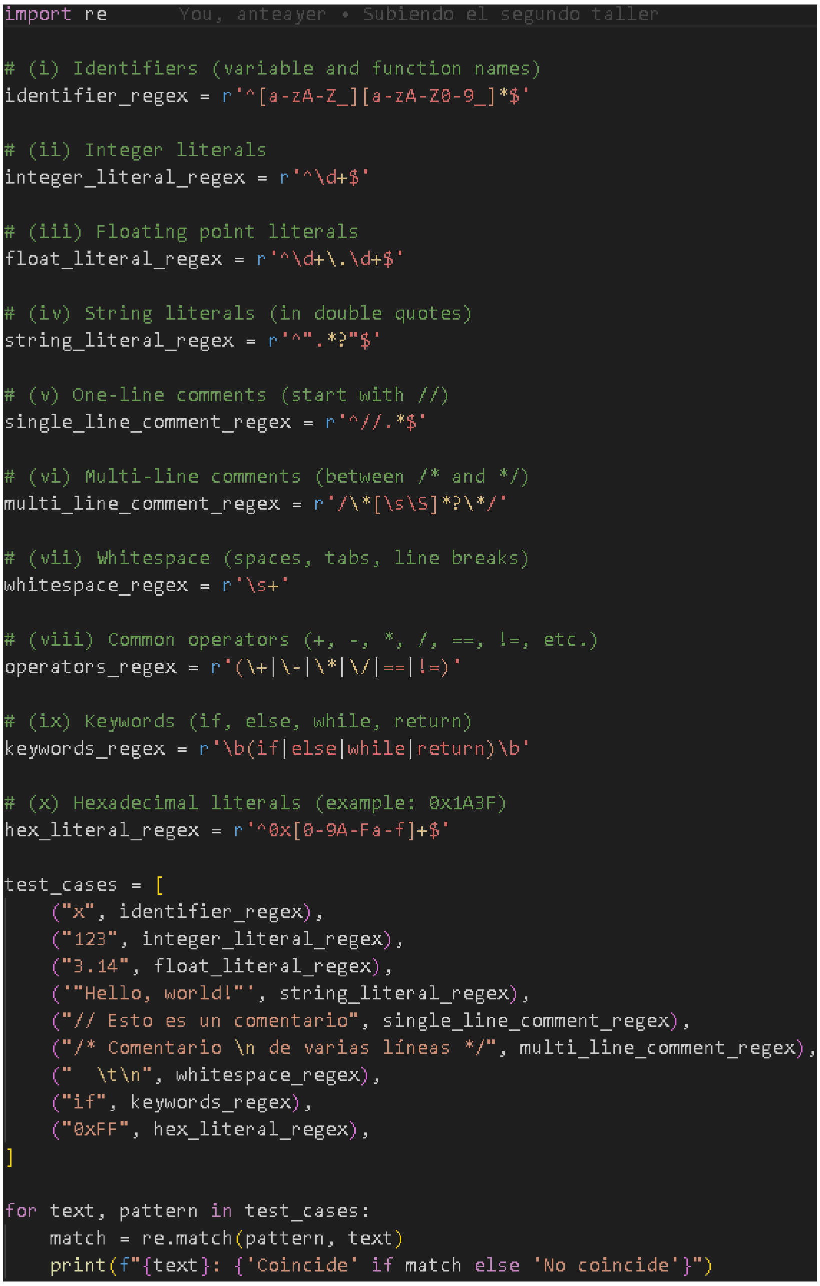
Faculty of Engineering

Computer Science III

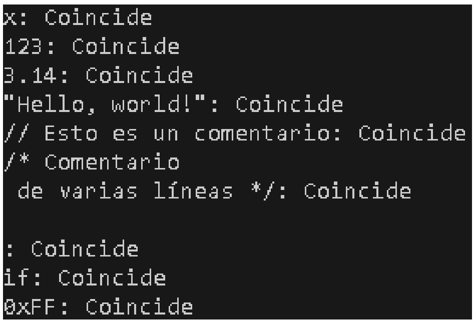
Bogotá, 2024

Exercices:

1. For each one of next cases definr a regular expression as used in a compiler based on the Python `re` library:



```python
import re       You, anteayer • Subiendo el segundo taller

# (i) Identifiers (variable and function names)
identifier_regex = r'^[a-zA-Z_][a-zA-Z0-9_]*$'

# (ii) Integer literals
integer_literal_regex = r'^\d+$'

# (iii) Floating point literals
float_literal_regex = r'^\d+\.\d+$'

# (iv) String literals (in double quotes)
string_literal_regex = r'^".*?"$'

# (v) One-line comments (start with //)
single_line_comment_regex = r'^//.*$'

# (vi) Multi-line comments (between /* and */)
multi_line_comment_regex = r'/\*[\s\S]*?\*/'

# (vii) Whitespace (spaces, tabs, line breaks)
whitespace_regex = r'\s+'

# (viii) Common operators (+, -, *, /, ==, !=, etc.)
operators_regex = r'(\+|\-|\*|\/|==|!=)'

# (ix) Keywords (if, else, while, return)
keywords_regex = r'\b(if|else|while|return)\b'

# (x) Hexadecimal literals (example: 0x1A3F)
hex_literal_regex = r'^0x[0-9A-Fa-f]+$'

test_cases = [
    ("x", identifier_regex),
    ("123", integer_literal_regex),
    ("3.14", float_literal_regex),
    ('"Hello, world!"', string_literal_regex),
    ("// Esto es un comentario", single_line_comment_regex),
    ("/* Comentario \n de varias líneas */", multi_line_comment_regex),
    ("  \t\n", whitespace_regex),
    ("if", keywords_regex),
    ("0xFF", hex_literal_regex),
]

for text, pattern in test_cases:
    match = re.match(pattern, text)
    print(f"{text}: {'Coincide' if match else 'No coincide'}")
```

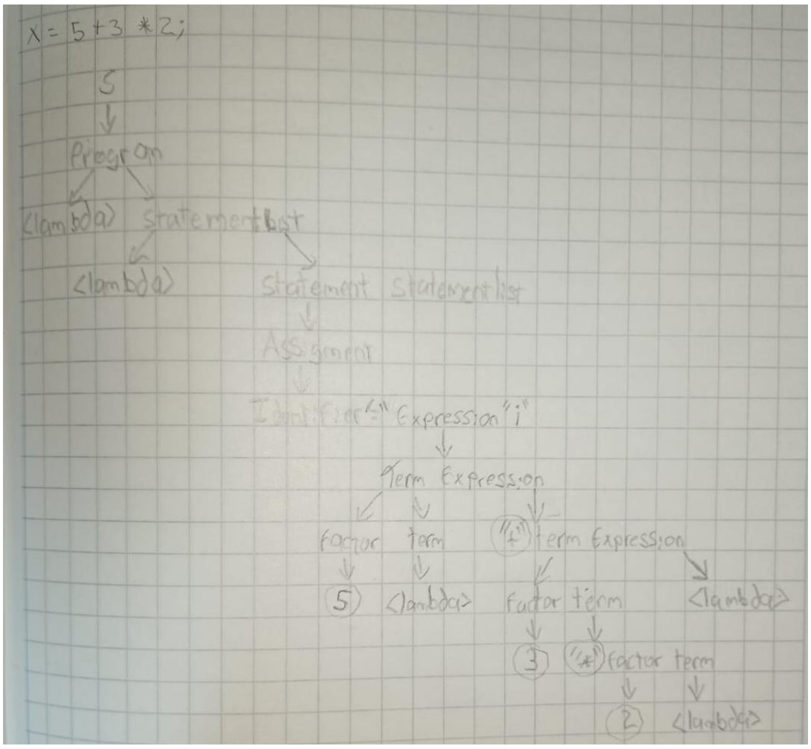


```
x: Coincide
123: Coincide
3.14: Coincide
"Hello, world!": Coincide
// Esto es un comentario: Coincide
/* Comentario
 de varias líneas */: Coincide

: Coincide
if: Coincide
0xFF: Coincide
```

2. Be G a `context-free grammar` with the following productions:

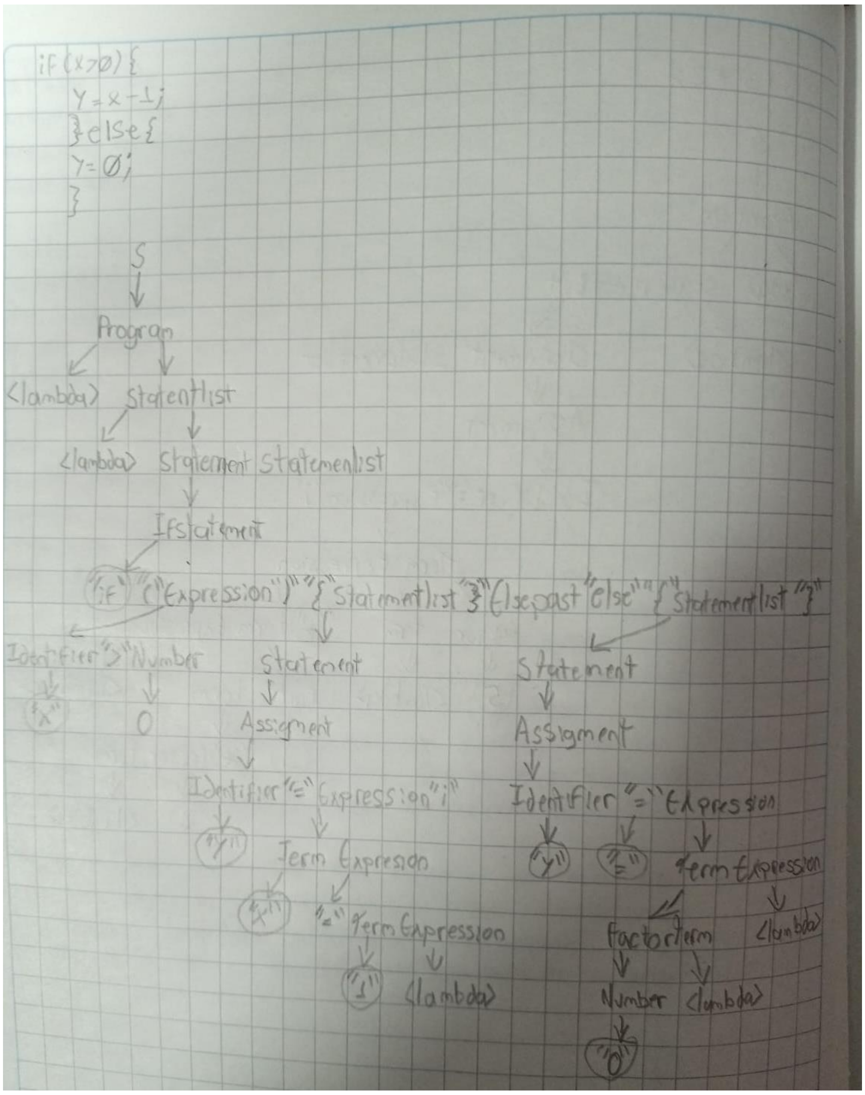
(a) **Exercise 1:**

```
x = 5 + 3 * 2;
```



(b) **Exercise 2:**

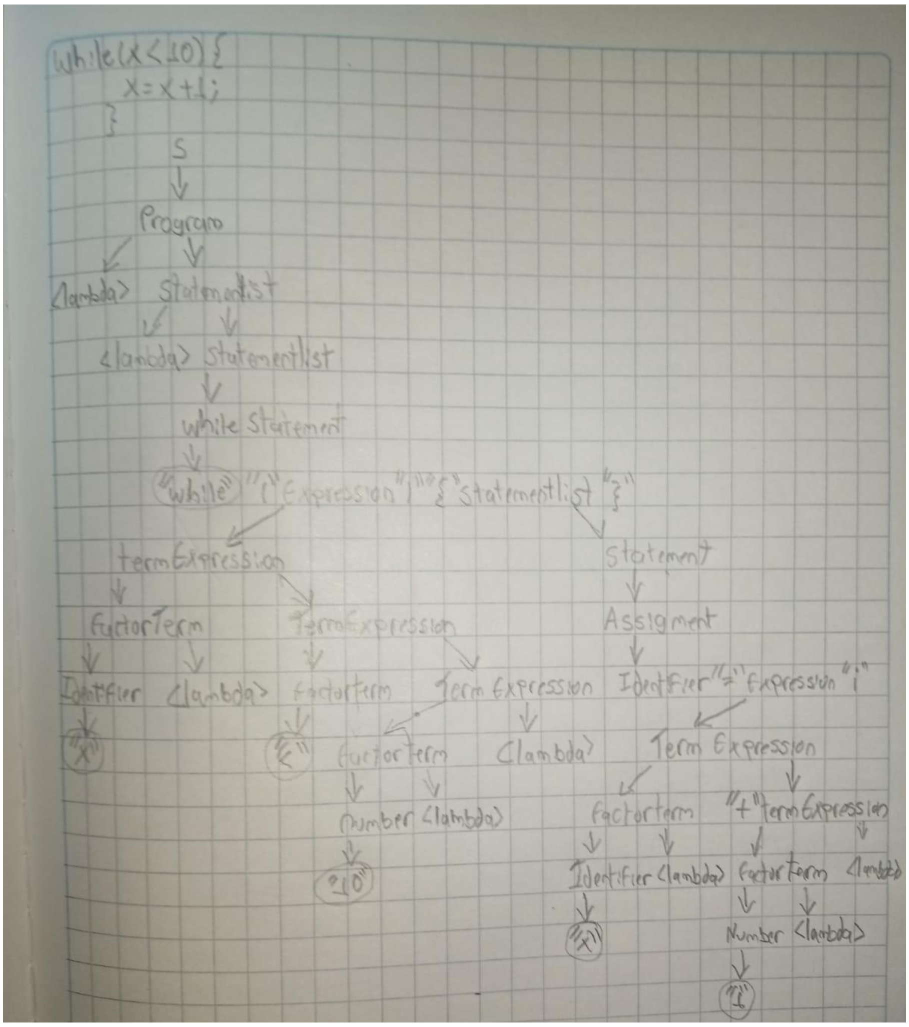
```
if (x > 0) {
    y = x - 1;
} else {
    y = 0;
}
```



```
if(x>Ø){
  Y=x-1;
  }else{
  Y=Ø;
  }
```

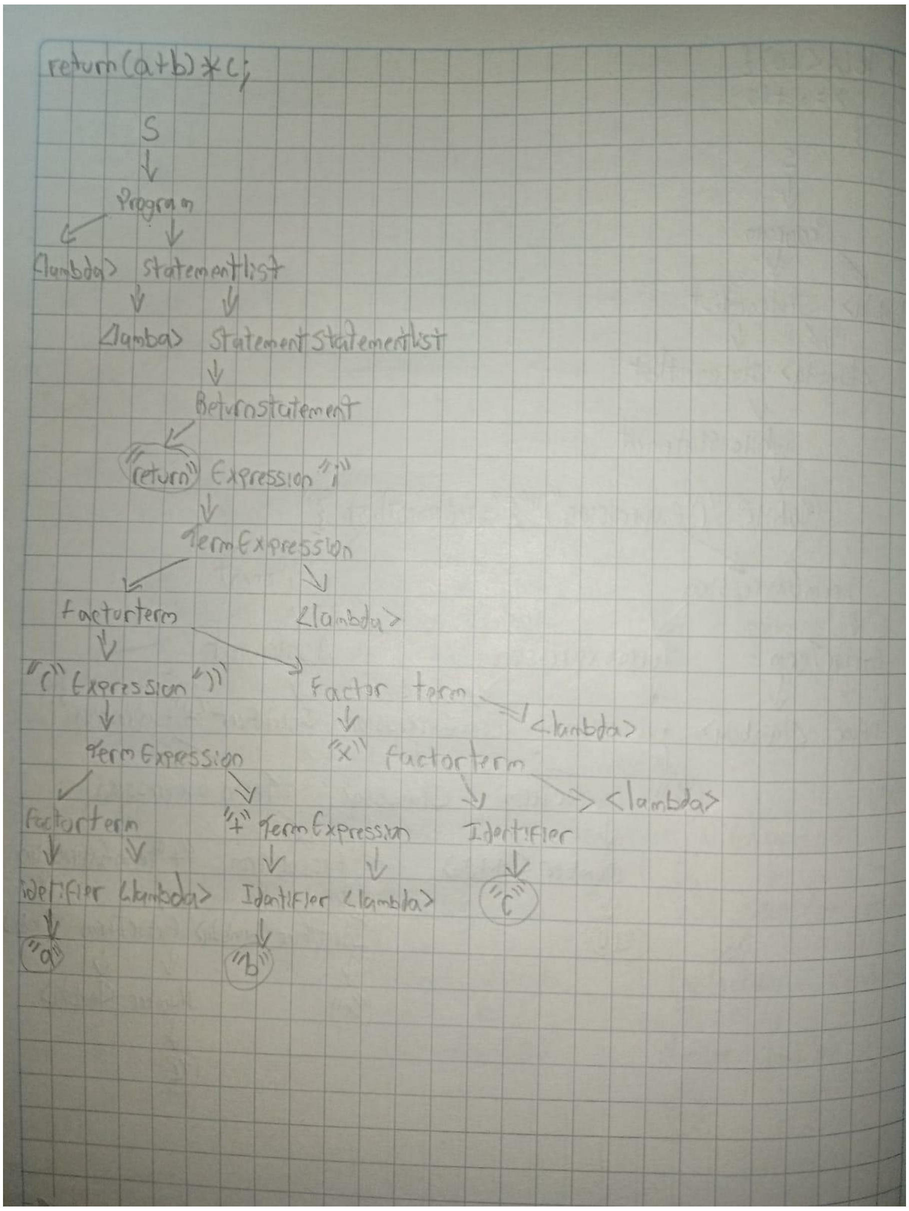
(c) **Exercise 3:**

```
while (x < 10) {
    x = x + 1;
}
```



(d) **Exercise 4:**

```
return (a + b) * c;
```


return(a+b)*c;
S
Program
<lambda>
Statementlist
<lamba>
Statement Statementlist
Returnstatement
"return"
Expression
";"
TermExpression
Factorterm
<lambda>
"("
Expression
")"
Factor
term
<lambda>
TermExpression
"*"
factorterm
<lambda>
Factorterm
"+"
TermExpression
Identifier
Identifier
<lambda>
Identifier
<lambda>
"c"
"a"
"b"