Manual Técnico Proyecto#02 - LFP

Compatibilidad

-Windows 10

-Linux

Tabla de tokens

Tabla

TOKEN	EXPRESIÓN REGULAR
REGISTROS	REGISTROS
CLAVES	CLAVES
max	max
min	min
average	promedio
count	count
parenthesis_that_opens	(
parenthesis_that_closes	)
Closed_key	}
Open_key	{
two_points	:
semicolon	;
comma	;
equals	=
quotation_mark	"
open_square_bracket	[
closed_square_bracket	]
int_value	[0-9]+
float_value	[0-9]+.[0-9]+

TOKEN	EXPRESIÓN REGULAR
count_if	contarsi
data	datos
print	imprimir
println	imprimirln
sum	sumar
export	exportarReporte
simple_comment	#
multiple_comment	'''

Gramática Utilizada

BEGGINING = INSTRUCTIONS_LIST

INSTRUCTIONS_LIST = INSTRUCTION INSTRUCTION_LIST'

INSTRUCTION = KEYS_INS
| REGISTERS_INS
| PRINT_INS
| PRINTLN_INS
| MAX_INS
| MIN_INS
| COUNTIF_INS
| EXPORT_INS
| SUM_INS
| DATA_INS
| AVERAGE_INS
| COUNT_INS

INSTRUCTIONS_LIST' = INSTRUCTION INSTRUCTIONS_LIST'
| Epsilon
| EOF -> sentinela

#ins registros:

REGISTERS_INS = register equal Open_square_bracket
REGISTERS_LIST closed_square_bracket

REGISTERS_LIST = REGISTER REGISTERS_LIST'

REGISTERS_LIST' = REGISTER REGISTERS_LIST''
| Epsilon
| closed_square_bracket (])

REGISTER = open_key ATTRIBUTES_LIST closed_key

ATTRIBUTES_LIST = ATTRIBUTE ATTRIBUTES_LIST'

ATRIBUTES_LIST' = comma ATTRIBUTE ATTRIBUTES_LIST'
| Epsilon (} closed_key)

ATTRIBUTE = string_value
| int_value
| float_value

#instrucción max:

MAX_INS = max parenthesis_that_opens parenthesis_that_closes
semicolon

#instrucción min:

MIN_INS = min parenthesis_that_opens parenthesis_that_closes
semicolon

#instrucción datas:

DATA_INS = data parenthesis_that_opens parenthesis_that_closes
semicolon

KEYS_INS = keys equal open_square_bracket KEYS_LIST closed_square_bracket # Instrucción claves:

KEYS_LIST = string_value KEYS_LIST'

KEYS_LIST' = comma string_value KEYS_LIST'
| epsilon

Instrucción imprimir:

PRINT_INS = print parenthesis_that_opens string_value parenthesis_that_closes semicolon

Instrucción imprimirln:

PRINTLN_INS = println parenthesis_that_opens string_value parenthesis_that_closes semicolon

INSTRUCCIÓN PROMEDIO:

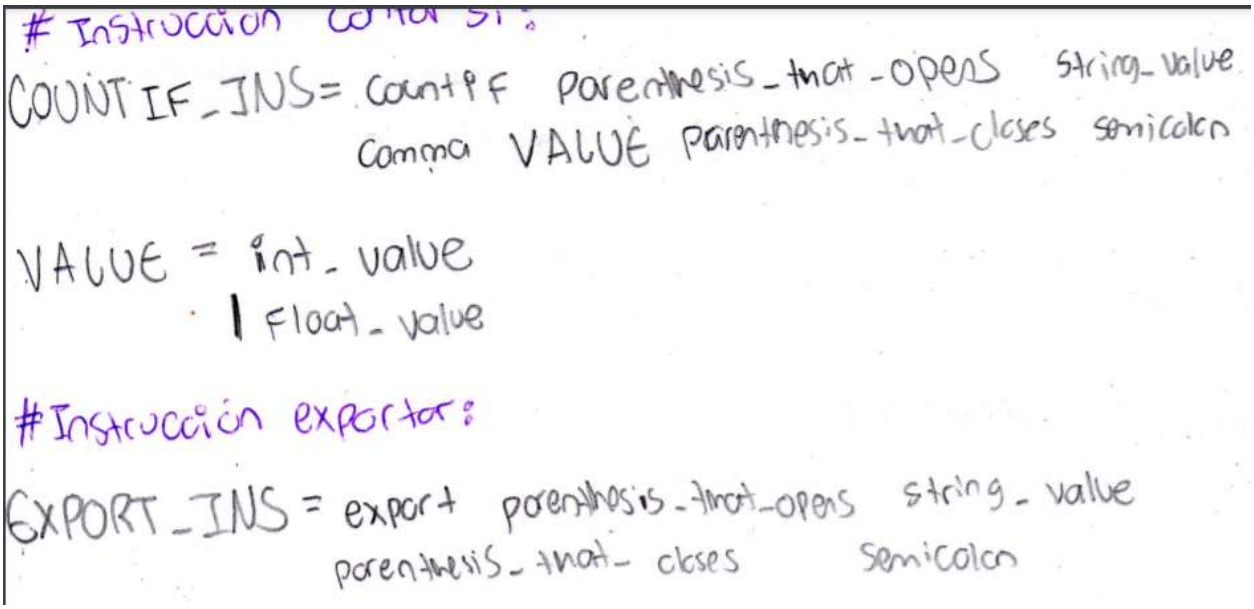
AVERAGE_INS = average parenthesis_that_opens string_value parenthesis_that_closes semicolon

Instrucción conteo:

COUNT_INS = count parenthesis_that_opens parenthesis_that_closes semicolon

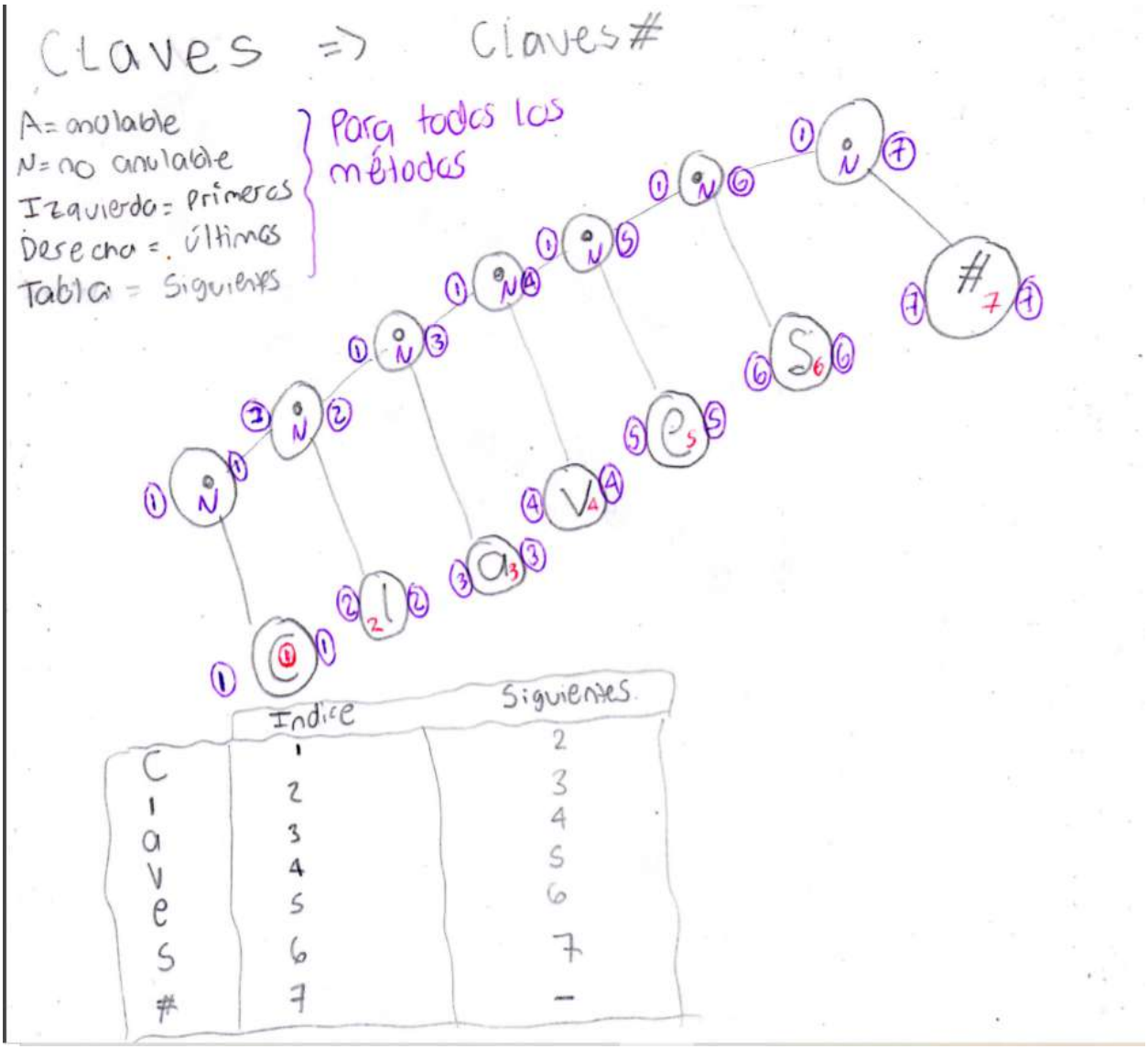
INSTRUCCIÓN Sumar:

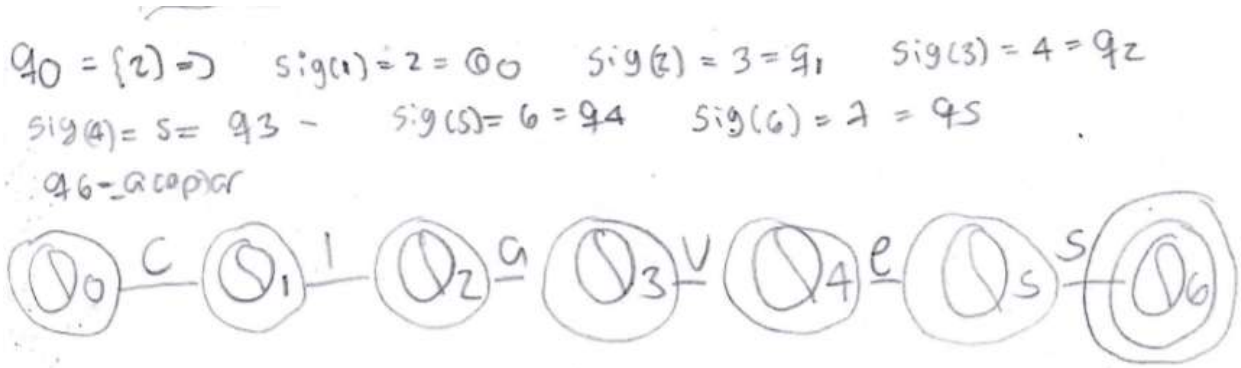
SUM_INS = sum parenthesis_that_opens parenthesis_that_closes semicolon



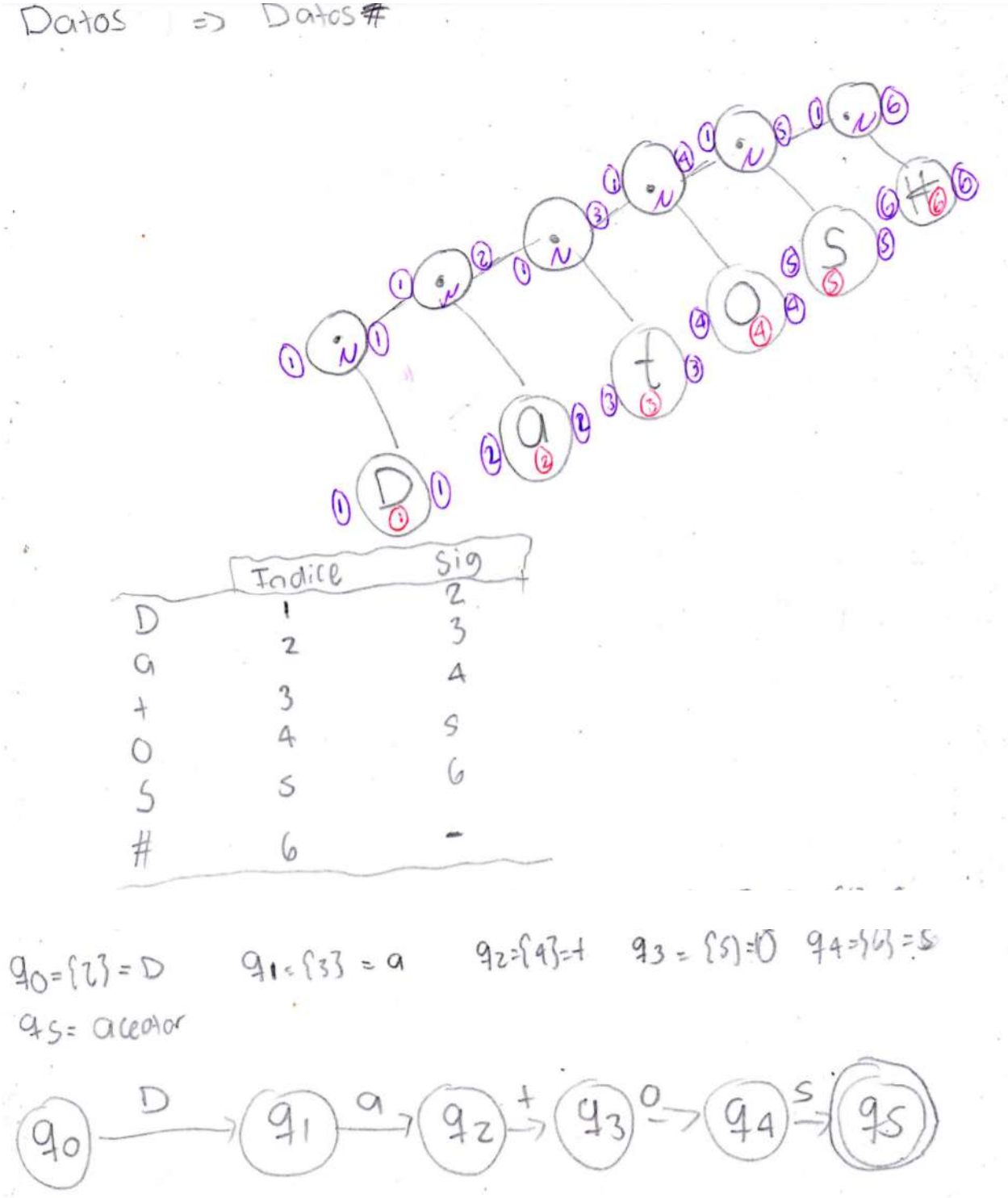
Métodos de Árbol y Autómatas

Claves

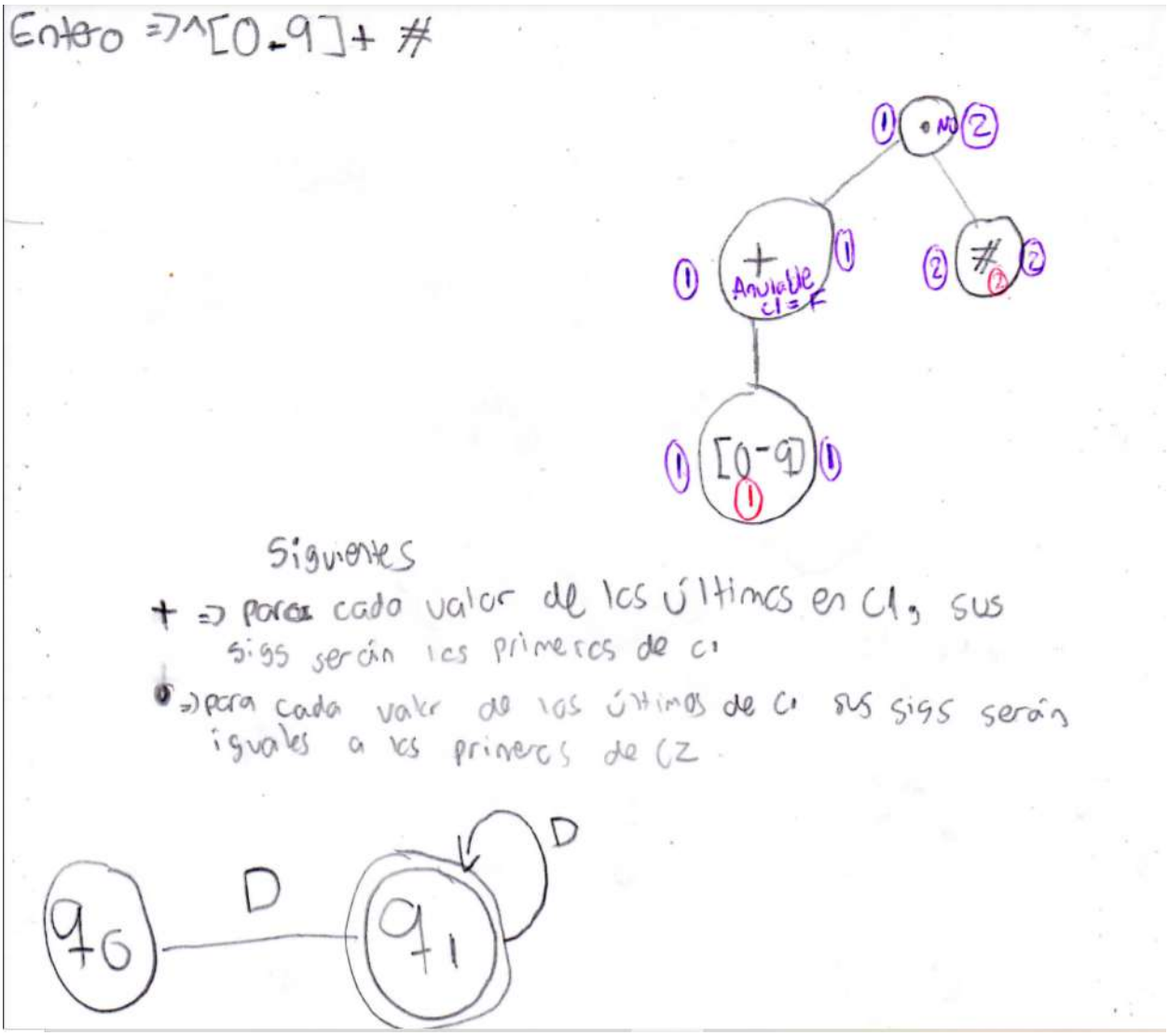

Claves => Claves#
A= anulable
N= no anulable
Izquierdo= primeros
Derecha = últimos
Tabla = Siguientes
Para todos los métodos
Indice
Siguientes
C 1 2
l 2 3
a 3 4
v 4 5
e 5 6
s 6 7
7 -


q0 = {2} => sig(1) = 2 = q0 sig(2) = 3 = q1 sig(3) = 4 = q2
sig(4) = 5 = q3 - sig(5) = 6 = q4 sig(6) = 7 = q5
q6 - aceptar
q0 C q1 l q2 a q3 v q4 e q5 s q6

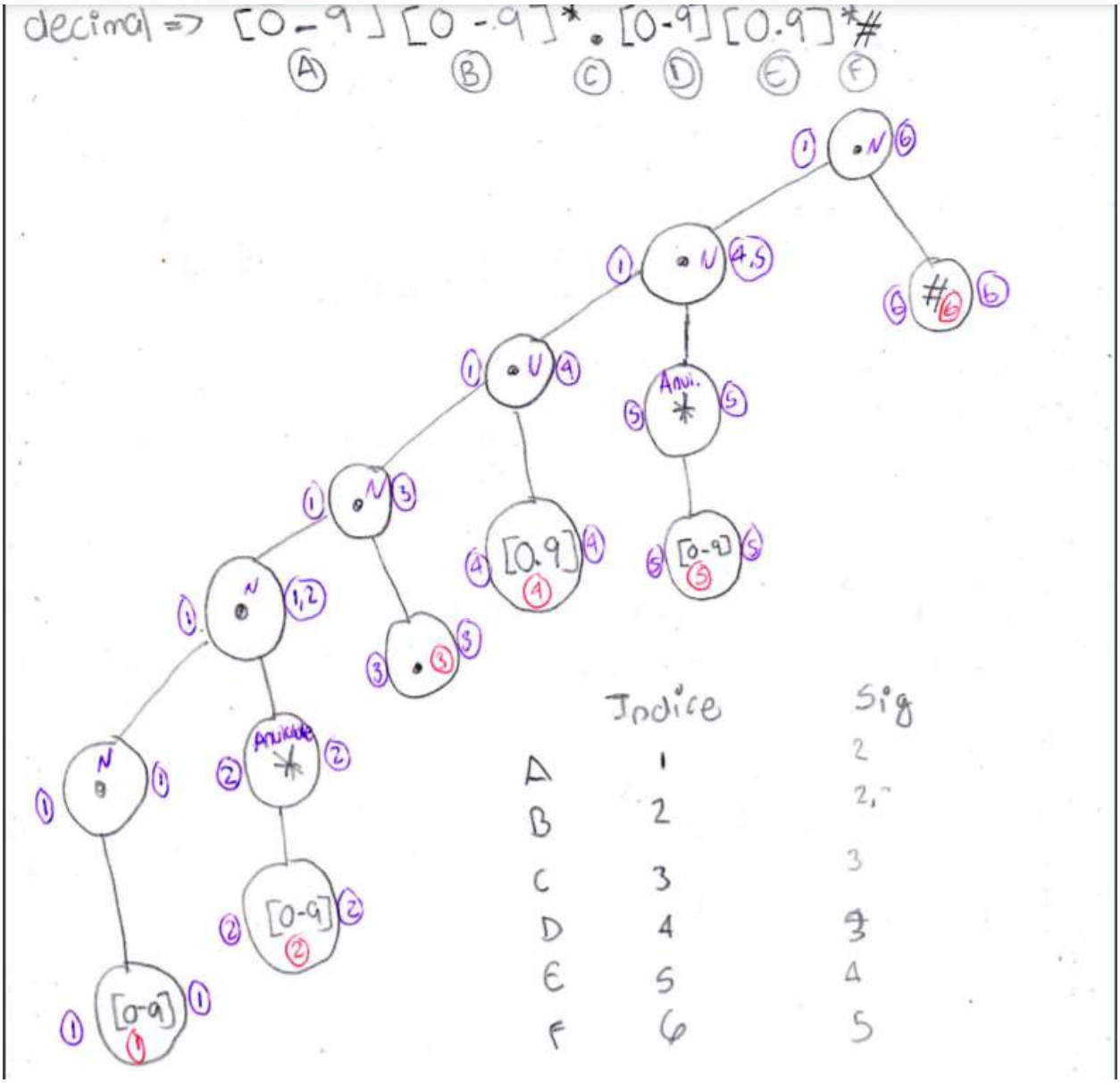
Datos

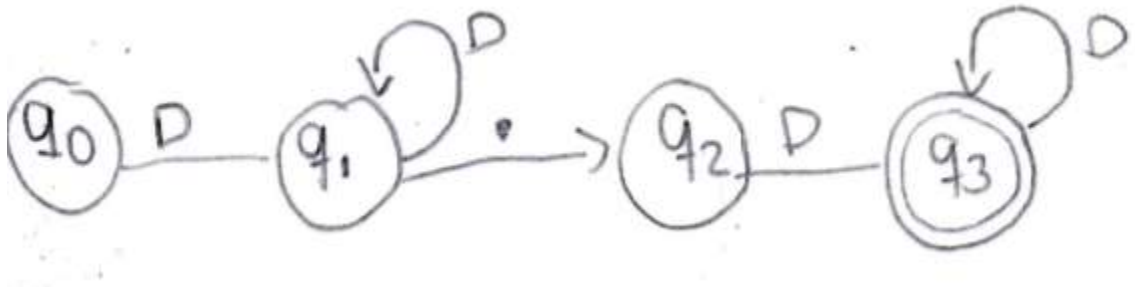


Número Entero



Número Decimal


decimal => [0-9] [0-9]* . [0-9] [0.9]* #
A B C D E F
Indice Sig
A 1 2
B 2 2,
C 3 3
D 4 3
E 5 4
F 6 5


q0
D
q1
D
.
q2
D
q3
D

Comentario

Comentario ⇒ #/^[^\n]*\n $

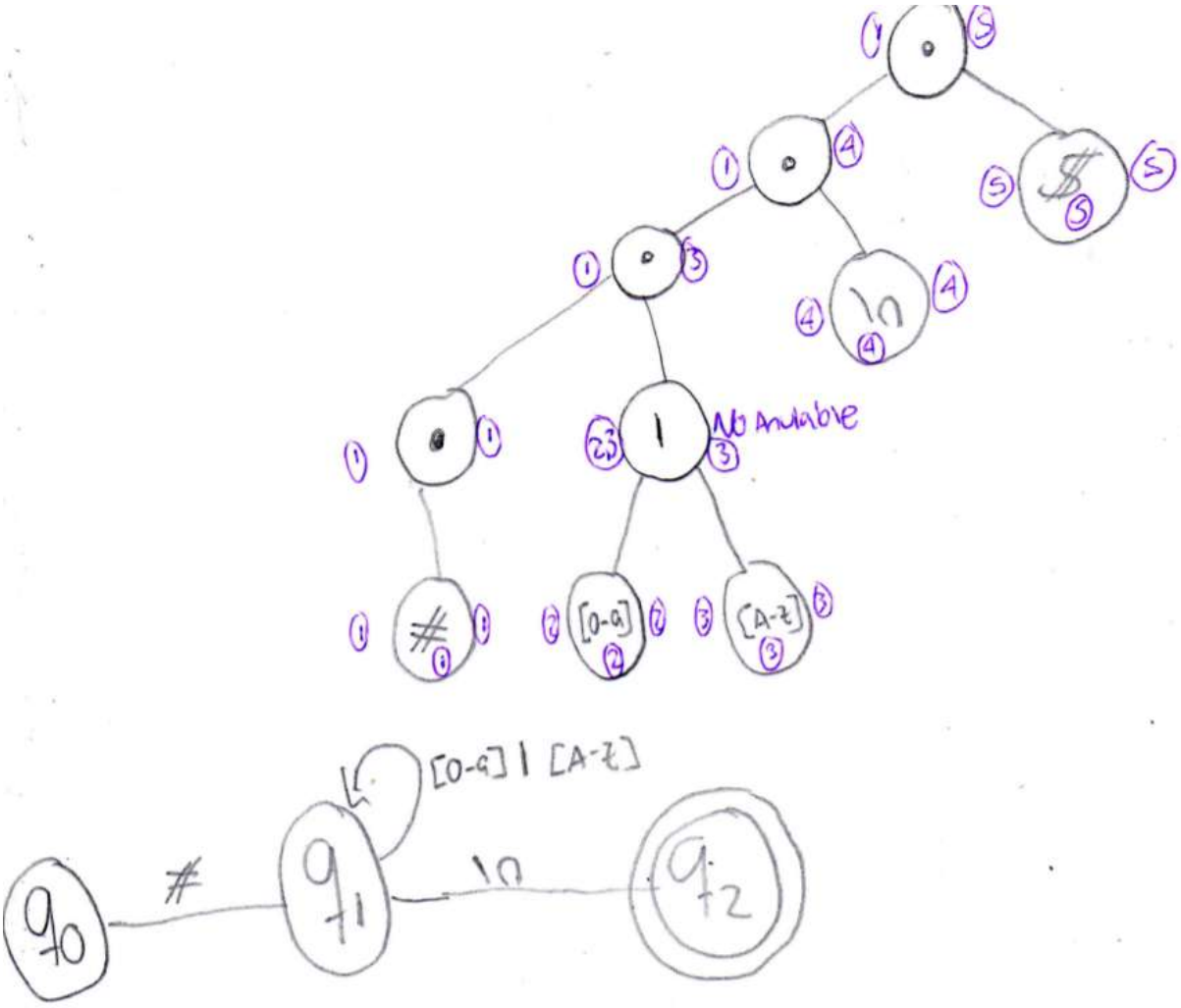
Explicación Regex:

/^ : Empieza con [^ : Que no sea "\n" en este caso

\n : caracter a excluir * : 0 ó más caracteres

\n : luego un salto de linea para terminar

$: sentinela



Funciones Utilizadas

Upload_file()

Función que se encarga de cargar el archivo de entrada y mostrar el texto en la consola de edición.

```
route=""
original_text=""
def upload_file():
    global route,original_text,editable_text
    route = askopenfilename()
    if route.endswith("lfp"):
        archive = open(route,"r")
        original_text=archive.read()
        archive.close()
        #print(original_text)
        editable_text.insert("1.0",original_text)
        messagebox.showinfo(title="Products Inventory Parser v1.0",
        message="El archivo fue cargado con éxito al sistema!!")
    else:
        messagebox.showerror(title="Products Inventory Parser v1.0",
        message="No es un archivo con extensión 'lfp', intenta de nuevo!!")
        upload_file()
upload_Button.config(command=upload_file)
```

Analyze_file()

Función encargada de recolectar el texto de la consola de edición para su posterior análisis

```
def analyze_file():
    global editable_text,console_text
    lexical_analyzer_handler.lexical_analyze_file(editable_text.get("1.0",'end-1c'))
    list=[]
    for token in token_handler.tokens_list:
        if token.type !="simple_comment" and token.type!="multiple_comment":
            list=[*list,token]
    parser_handler.analyze(list)
    console_text.configure(state='normal')
    console_text.insert("end-1c",register_handler.report_console)
    console_text.configure(state='disabled')
analyze_Button.config(command=analyze_file)
```

student_information()

Función encargada de mostrar la información del estudiante

```
def student_information():
    messagebox.showinfo(title="Products Inventory Parser v1.0",
    message="Master Mind: Erwin Vásquez\n LFP Sección 'B+'\n Proyecto 2")
information_Button.config(command=student_information)
```

reports_view()

Función encargada de abrir los reportes de tokens y errores.

```
def reports_view():
    token_handler.tokens_html_report()
    error_handler.errors_html_report()
    messagebox.showinfo(title="Image Maker V1.0",
    message="Se Abrirán Los siguientes Reportes:\n-Reporte De Tokens\n"
    "-Reporte De errores")
    os.system("C:/Users/Erwin14k/Documents/Products_Inventory_Parser/REPORTES/TOKENS.html")
    os.system("C:/Users/Erwin14k/Documents/Products_Inventory_Parser/REPORTES/ERRORS.html")
reports_Button.config(command=reports_view)
```

exit_aplication()

Función encargada de darle fin a la ejecución de la aplicación.

```
def exit_application():
    exit()
exit_Button.config(command=exit_application)
```