socrative

PLC21-Exe3

5 Questions

---

1. Considere as 3 seguintes frases válidas de uma linguagem L

Ana(-, Rui (Maria (-,-), Joao(-, -)))
Jose(Marta (-,-), Antonio (-,-))
Sara(Rita (Maria(-,-), Tomas (-,Vasco)), -)

Escolha então nas alíneas seguintes as GIC que podem gerar a linguagem L

30/44 **A**

```
ArvGen : GenT
GenT   : '-'
GenT   : name '(' GenT ',' GenT ')'
```

29/44 **B**

```
ArvGen : id PE ArvGen VI ArvGen PD
       | '-'
PE     : '('
VI     : ','
PD     : ')'
```

30/44 **C**

```
ArvGen : GenT
       | ArvGen GenT
GenT   : '-'
GenT   : name '(' GenT ',' GenT ')'
```

2/44 **D**

```
ArvGen : GenT ','
GenT   : '-'
GenT   : GenT '(' GenT ',' GenT ')'
```

2. Considere a seguinte GIC geradora de números binários

$$S \rightarrow 0\,S\,1\,S \mid 1\,S\,0\,S \mid \epsilon$$

Das frases seguintes assinale aquelas que estão corretas de acordo com a gramática:

12/42 **A** **100101101**

15/42 **B** **00110001**

34/42 **C** **01**

26/42 **D** **00001111**

3. Considere a seguinte GIC

Sequencia : Intervalos
Intervalos : €
Intervalos : Intervalos Intervalo
Intervalo : '[' NUM Sep NUM ']'
Sep : ',' | ':'

e selecione as afirmações seguintes que são verdadeiras

41/43 **A** A frase **[1,2] [4:1] [2:5] [5,6]** pertence à linguagem gerada pela GIC

36/43 **B** A frase **vazia** pertence à linguagem gerada pela GIC

3/43 **C** A frase **[1,2] [4:5,12:15] [5,6]** pertence à linguagem gerada pela GIC

7/43 **D** A frase **[[1,2] , [4:1]]** pertence à linguagem gerada pela GIC

4. Considere o seguinte Analisador Léxico (ALex) em Python

```
import ply.lex as lex
import sys
tokens = ( 'LPAREN' , 'RPAREN' , 'VIRG' , 'PT' ,
           'DP' , 'TS' , 'AL' , 'NTS' , 'ID' , 'NUM' )
t_LPAREN = r'\('
t_RPAREN = r'\)'
t_VIRG   = r','
t_DP     = r':'
t_PT     = r'\.'
t_TS     = r'(?i:TURMAS)'
t_AL     = r'(?i:ALUNO)'
t_NTS    = r'(?i:NOTAS)'
t_ID     = r'\w+'
def t_NUM(t):
  r'\d+'
  t.value = int(t.value)
  return t
t_ignore = ' \r\n\t'
def t_error(t):
  print('Illegal character: ' + t.value[0])
  t.lexer.skip(1)
alex = lex.lex()
for linha in sys.stdin:
  alex.input(linha)
  tok = alex.token()
  while tok:
    print(tok)
    tok = alex.token()
```

E selecione as afirmações verdadeiras

33/44 **A** pode dizer-se que '**ply.lex**' é um processador que lê a especificação no topo do programa e gera um objeto (nesta caso 'alex') que é um Analisador Léxico.

34/44 **B** o construtor '**t_ignore**' acima serve para listar os carateres da entrada que são aceites mas que são desprezados sem afetar o processamento.

10/44 **C** se o analisador léxico ler da entrada um carater não pertencente a nenhum dos símbolos declarados, é emitida uma mensagem de erro e o programa termina.

21/44 **D** se a função '**t_NUM**' definida a partir de '**def t_NUM(t):**' fosse substituída pela pela declaração de linha **t_NUM = r'\d+'** o programa passaria a ter um comportamento diferente.

5. Considere o seguinte Analisador Léxico (ALex) em Python

```
import ply.lex as lex
import sys
states = (('comentario','inclusive'),)
tokens = ( 'CON', 'COFF', 'COM', 'PAL', 'NUMBER' )
t_NUMBER = r'\d+'
t_PAL    = r'[a-zA-Z]+'
def t_CON(t):
   r'/\*'
   t.lexer.begin('comentario')
   #return(t)
t_ignore  = ' \t\r\n'
def t_error(t):
   print("ERRO")
   t.lexer.skip(1)
def t_comentario_COFF(t):
   r'\*/'
   t.lexer.begin('INITIAL')
   #return(t)
def t_comentario_COM(t):
   r'.|\n'
lexer = lex.lex()
for linha in sys.stdin:
   lexer.input(linha)
   tok = lexer.token()
   while tok:
      print(tok)
      tok = lexer.token()
```

e selecione as afirmações verdadeiras:

17/43 **A** o ALex não vai funcionar porque as instruções '**return**' não podem estar comentadas.

18/43 **B** face ao texto de entrada **wdfb 345 /*kkjf-.,2535*/ 66** o ALex retorna uma **PAL** e dois **NUMBER** e elimina o comentário.

11/43 **C** face ao texto de entrada **wdfb 345 /*kkjf-.,2535*/ 66** o ALex retorna uma **PAL** e dois **NUMBER** e ainda os símbolos **CON , COFF** e **COM**.

15/43 **D** face ao texto de entrada **wd-.,o34545 /*kkjf2535lixo*/ fim**
o ALex retorna três **PAL** e um **NUMBER** e três **ERRO**

22/43 **E** caso retirasse o comentário dos dois 'return', o ALex ao ler **/* ola 12 */**" retornava exatamente **CON** e **COFF**