socrative
A showbie PRODUCT

PLC22-mT2

10 Questions

1. Selecione a alínea abaixo que é uma afirmação verdadeira:

2 POINTS

10/36 **A** Um Compilador, após analisar completamente a frase de entrada e caso esta seja válida à luz da gramática da respetiva linguagem, executa de imediato as ações que ela preconiza.

15/36 **B** Um Filtro de Texto analisa e transforma o texto de entrada sem ter de verificar se é uma frase válida à luz da gramática da respetiva linguagem.

9/36 **C** Um Filtro de Texto baseia-se na sintaxe da linguagem de entrada para executar a sua tarefa.

2/36 **D** Um Interpretador recebe uma frase e executa de imediato as ações que ela preconiza sem ter de verificar que seja válida à luz da gramática da respetiva linguagem

2. Selecione a alínea abaixo que é uma afirmação verdadeira:

2 POINTS

19/36 **A** Um Analisador Sintático reconhece a forma ou estrutura do programa de entrada e representa-a numa árvore.

1/36 **B** Um Parser reconhece os símbolos terminais da linguagem.

11/36 **C** Um Analisador Sintático analisa o tipo das variáveis e das expressões e avalia a sua concordância.

5/36 **D** Um Analisador Léxico constrói uma árvore de sintaxe que representa o programa fonte e sobre a qual trabalham o Gerador e Otimizador de código.

3. Relembre o que sabe sobre Autómatos Não-deterministas (AND) e Deterministas (AD) e Expressões Regulares (ER),
e selecione a alínea abaixo que é uma afirmação verdadeira:

2 POINTS

8/37 **A** Nem sempre é possível converter um AND em AD.

2/37 **B** É sempre possível transformar formalmente num só passo uma ER num AD.

6/37 **C** Por construção, o autómato que se obtém a partir de uma ER por aplicação das regras formais conhecidas, é determinista.

21/37 **D** Para converter sistematicamente um AND em AD, deve começar-se por lançar mão de uma função dita €-fecho que elimina todos os caminhos de peso epsilon.

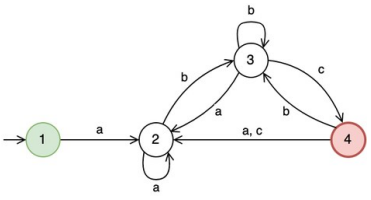
4. Ajuíze a veracidade da seguinte afirmação
« *Transformando um conjunto de ER num AD é possível implementar um algoritmo iterativo genérico, muito rápido e pequeno (com 3 instruções elementares e uma estrutura de controlo), cujo comportamento é determinado pela função de transição do dito AD, para obter um programa que é capaz de reconhecer qualquer 'string' derivada de uma dessas ER.* »

2 POINTS

30/36 **T** True

6/36 **F** False

5. Observe o autómato **A1** da figura ao lado (**4** é um estado final).



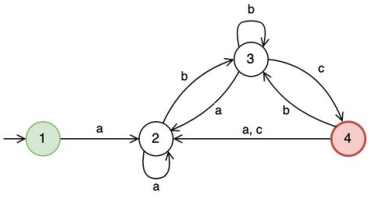
Diga, então, se a seguinte afirmação é verdadeira ou falsa:
« *O autómato **A1** é equivalente à seguinte expressão regular:*
a(a|b|c)*bc
»

2 POINTS

3/33 **T** True

30/33 **F** False

6. Observe o autómato **A1** da figura ao lado (**4** é um estado final).



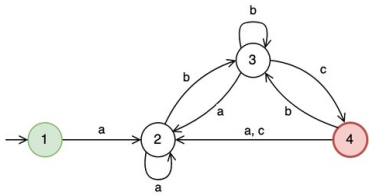
Diga, então, se a seguinte afirmação é verdadeira ou falsa:
« *A Tabela Delta que representa a função de transição do autómato **A1** tem 4 linhas e 4 colunas (considerando que o conjunto dos símbolos terminais inclui o '$', EOF) e das suas 16 entradas, 6 vão para erro.* »

2 POINTS

21/35 **T** True

14/35 **F** False

7. Observe o autómato **A1** da figura ao lado (**4** é um estado final).



Selecione, então, a alínea abaixo que é uma afirmação verdadeira:

2 POINTS

2/37 **A** O autómato **A1** é Não-determinista porque há 3 ramos '**b**' a entrar no mesmo estado **3**.

2/37 **B** O autómato **A1** é Não-determinista por ter 3 ramos de saída a partir de **4,** que é um estado final.

7/37 **C** O autómato **A1** é Não-determinista por ter uma transição (de **4** para **2**) correspondente a 2 símbolos distintos.

26/37 **D** Apesar de ter 4 ramos de entrada no estado **2**, etiquetados com o símbolo '**a**', o autómato **A1** é Determinista.

8. Considere o seguinte extrato de um filtro de texto em Python

```
import re
linha = input()
y = re.findall(r'[ ][^0-9]+\.', linha)
if(len(y)>0):
    print("existem ", len(y), " ocorrências, a primeira ", y[0])
else:
    pass
```

e selecione a alínea abaixo que é uma afirmação verdadeira:

2 POINTS

15/35 **A** se o texto de entrada for

3 hh-+.345.ola.12. 34 rrr.1.rnhh .89. ghhh .

a resposta do programa é

existem 3 ocorrências, a primeira: ' hh-+.'

6/35 **B** se o texto de entrada for

ola. 12. 34 rrr.1.rnhh .89. ghhh .3 hh-+.345."

a resposta do programa é

existem 5 ocorrências, a primeira: 'ola. '

6/35 **C** se o texto de entrada for

ola.12.34. rrr.1.rnhh . 89. ghhh .3 hh-+.345.

a resposta do programa é

existem 5 ocorrências, a primeira: '12.'

8/35 **D** se o texto de entrada for

ola.12. 34 rrr.1.rnhh .89. ghhh .3 hh-+.345.

a resposta do programa é *vazia* (não escreve nada).

9. Considere o seguinte extrato de um filtro de texto em Python

```
import re
import sys
for linha in sys.stdin:
  if ( s := re.search(r'\[([aeiou]+|[13579]*)\]', linha) ):
    print(s.group())
  else:
    print("falhou!")
```

e selecione a alínea abaixo que é uma afirmação verdadeira:

2 POINTS

3/36 **A** se o texto de entrada for
LINHA COM 1 marca de SUCESSO (a) ou (2)
a resposta do programa é "**2**".

21/36 **B** se o texto de entrada for
LINHA COM 1 marca de SUCESSO (a) ou (2)
a resposta do programa é "**falhou!**".

5/36 **C** se o texto de entrada for
LINHA COM [1] marca de SUCESSO (a) ou (2)
a resposta do programa é "**1**".

7/36 **D** se o texto de entrada for
LINHA COM [] marcas de SUCESSO [a] ou [13]
a resposta do programa é "**[a]**".

10. Considere o seguinte excerto de um analisador léxico:

```
import ply.lex as lex
tokens = ( 'ID', 'ARG', 'STR', 'PONTO' )
t_PONTO = r'.'
t_ID      = r'\w+'
t_ARG     = r'\$\d+'
t_STR     = r'\"[^\"]*\"'
t_ignore = ' \n\t'
def t_error(t): .....
```

Assinale, então, a afirmação FALSA:

2 POINTS

19/39 **A** No input

print . $1 . match "hello"

se se substituir o carater "." por "," ou por "|" ,
o resultado do Analisador será diferente visto que o token **PONTO** só reconhece o carater "."

1/39 **B** Ao processar a frase

print . $1 . match "hello"

o Analisador reconheceria 6 símbolos terminais.

12/39 **C** Ao processar a frase

Bem vindo de volta, $UTILIZADOR!

o Analisador reconheceria 5 símbolos ID.

7/39 **D** Na frase

Bem vindo de volta, $UTILIZADOR!

se a palavra 'UTILIZADOR' fosse substituída por 23, o Analisador reconheceria ao todo 7 símbolos terminais.