Exame de época normal: pl2025-recurso

UC: Processamento de Linguagens (LEI)

14 de Junho de 2025, 9h30-12h30,

Engenharia Informática (3° ano)

Questão 1: Expressões Regulares (especificação) (3 val.)

Escreva uma expressão regular que "apanhe":

- **a)** (1 val.) Dado um ficheiro de registo de logs (log file), apanhe todas as linhas que começam com a data no formato `YYYY-MM-DD`;
- **b)** (1 val.) Dado um documento HTML, apanhe todas as tags `<img>` que possuem o atributo `alt`:
  - Exemplo entrada:

```
<img alt="foto pessoal" src="f1.jpg"/>
<img width="4cm" alt="foto pessoal" src="f2.jpg"/>
<img src="f1.jpg" align="top"/>
```

  - Exemplo saída:

```
<img alt="foto pessoal" src="f1.jpg"/>
<img width="4cm" alt="foto pessoal" src="f2.jpg"/>
```

- **c)** (1 val.) Dado um programa Python, apanhe linhas "`for ....`" que não terminem com "`:`".

Questão 2: Expressões Regulares (módulo re) (3 val.)

Considere o seguinte programa Python, no qual uma expressão regular é usada para extrair informações de uma string formatada:

```
import re

def process_data(text):
    pattern = r"ID:(\d+)\s+Nome:([A-Za-z\s]+)\s+Valor:(\d+\.\d{2})"
    match = re.search(pattern, text)
    if match:
        return {
            "id": int(match.group(1)),
            "nome": match.group(2).strip(),
            "valor": float(match.group(3))
```

```
        }
    else:
        return None

data_string = "ID:1234 Nome:Produto A Valor:99.99"
result = process_data(data_string)
print(result)
```

- **a)** (1 val.) Apresente o resultado produzido pelo programa;
- **b)** (1 val.) Escreva um `re.findall(REGEXP,texto_html)` que capture a lista dos links (`<a ...>`) contidos num texto HTML em formato `list(tup(url,texto-de-desc))`

Exemplo:

```
texto_html: <a href="url-1">home</a> .... <a href="url2">contactos</a>
resultado: [ ("url-1", "home"), ("url2","contactos") ]
```

- **c)** (1 val.) Escreva um `re.sub(...)` que dada uma frase, lista de palavras separadas por espaço ou caráter não alfanumérico, para cada palavra deixe apenas o caráter inicial:
  - Exemplo: entrada - `era uma vez`; saída - `e u v`

Questão 3: Gramáticas (5v)

Escreva, em BNF-puro, uma Gramática Independente de Contexto (GIC) para definir formalmente uma linguagem específica para descrever um **menu de restaurante digital**, de acordo com as seguintes definições:

- O menu é dividido em categorias (e.g., entradas, pratos principais, sobremesas, bebidas);
- Cada categoria possui um título e uma lista não-vazia de itens;
- Cada item do menu tem um nome, uma descrição opcional e um preço;
- O menu deve começar com a indicação do dia da semana (**domingo**, **segunda**, ..., **sabado**) e a indicação do prato do dia com o respetivo preço. Note que o preço deve ser um número decimal e que o título, a descrição e o nome do item devem permitir escrever um texto qualquer entre parentesis retos.

Como a ideia é construir um parser *TopDown*, a GIC a escrever não deve ter conflitos LL(1). Prove-o no fim de a especificar (1,5 val.).

Questão 4: Compilador (6 val.)

A gramática independente de contexto, abaixo escrita em *BNF*, define uma linguagem de domínio específico para descrição de um conjunto de **comandos para um robô**.

O Símbolo Inicial é `ListaComandos`, os Símbolos Terminais são escritos entre apóstrofes se forem sinais, só em maiúsculas se forem palavras-reservadas (e.g., `MOVER`, `RODAR`, `AGUARDAR`), ou só em minúsculas se

forem terminais-variáveis (como: `int` que representa um número inteiro com sinal); os restantes (sempre começados por maiúsculas) serão os Símbolos Não-Terminais.

```
Z ::= ListaComandos '$'
ListaComandos ::= Comando Comandos
Comandos ::= ε | ';' ListaComandos
Comando ::= MOVER distancia | RODAR angulo | AGUARDAR tempo
distancia ::= int
angulo ::= int
tempo ::= int
```

Neste contexto e após analisar a gramática apresentada, responda às alíneas seguintes:

- **a)** (1 val.) Escreva uma *frase válida* da linguagem gerada pela gramática, apresentando a respetiva *árvore de derivação*;
- **b)** (2 val.) Escreva, em linguagem algorítmica, as funções de um parser RD (recursivo-descendente) apropriada para reconhecer as frases da linguagem gerada pela gramática;
- **c)** (1 val.) Escreva em Python usando o módulo `ply.lex` um analisador léxico para reconhecer as frases da linguagem definida pela gramática;
- **d)** (2 val.) Escreva em Python usando o módulo `ply.yacc` um analisador sintático para reconhecer as frases da linguagem definida pela gramática. Acrescente Ações Semânticas às produções da gramática para **calcular o número total de comandos de avanço (`MOVER`)** e **uma lista com todos os valores de `tempo` utilizados nos comandos `AGUARDAR`**.

Questão 5: Assembly da VM (3 val.)

Considere o seguinte programa escrito em Assembly da máquina virtual VM:

```
   PUSHN 6
   PUSHI 0
   PUSHI 1
   PUSHI 0
   START
   JUMP main

f1: NOP
   PUSHG 8
   PUSHG 8
   MUL
   WRITEI
   RETURN

main: NOP
   PUSHI 5
   STOREG 8
   PUSHA f1
   CALL
   PUSHI 3
```

```
    STOREG 8
    PUSHA f1
    CALL
    STOP
```

Responda, então, às alíneas seguintes:

- **a)** Quantas variáveis inteiras são reservadas na memória global ?
- **b)** Escreva um algoritmo que descreva o comportamento do programa;
- **c)** Para cada afirmação seguinte, diga justificando se é verdadeira ou falsa:
    1. O programa carrega para um array de dimensão 6, os 6 primeiros números naturais.
    2. Quando executado o programa imprime os números 25 e 9.
    3. Quando executado o programa imprime o número 25 e termina de imediato.
    4. Se a segunda e terceira instruções fossem:

```
    PUSHI 20
    PUSHI 15
```

o resultado do programa seria diferente.

Bom trabalho e boa sorte A equipe docente