UNIVERSIDADE AUTÓNOMA DE LISBOA LUÍS DE CAMÕES
Departamento de Ciências e Tecnologias
Algoritmia e Programação 2020/2021
Laboratório 4

Objetivos

Neste laboratório pretende-se praticar com todas as estruturas de dados introduzidas em AP, nomeadamente, listas, tuplos, e dicionários.

Datas

- Enunciado: 15 de dezembro de 2020
- Entrega no *e-learning*: 10 de janeiro de 2021

Regras

O trabalho deve ser realizado por grupos de 4 elementos, previamente registados no *e-learning*.

A entrega do trabalho é feita no *e-learning*. Deve ser entregue um ficheiro zip com o seguinte conteúdo:

- Diretório **src** com todo o código fonte;
- Diretório **doc** com eventuais diagramas e outra documentação que considerar relevante.

A entrega não será válida se for utilizado um formato de compressão além do zip, ou se não for possível abrir o ficheiro.

Tarefas

Parte 1 (4 valores)

Considere o ficheiro `nomes.csv` com informação dos nomes registados no ano de 2016 em Portugal Continental e Madeira (Fonte: `https://www.publico.pt/2017/01/03/sociedade/noticia/santiago-destrona-joao-num-pais-de-marias-1756999`)
Implemente um módulo `nomes.py`, com as seguintes funções:

`ler_nomes(nome_ficheiro)` Consulta o ficheiro e retorna uma lista de dicionários. Cada dicionário tem a seguinte estrutura:

```
{
    'nome': str,
    'registos': int,
    'genero': str
}
```

`listar_nomes(nomes)` Escreve na linha de comandos todos os nomes na lista de dicionários `nomes`.

`listas_nomes_genero(nomes, genero)` Escreve na linha de comandos todos os nomes da lista de dicionários `nomes`, de um determinado género.

`listas_nomes(nomes, min_registos)` Escreve na linha de comandos todos os nomes com um número de registos igual ou superior a `min_registos`, na lista de dicionários `nomes`.

Parte 2 (4 valores)

Pretende-se construir o jogo *Minesweeper*. Este é um jogo de computador muito popular que funciona sobre um campo de bombas espalhadas aleatoriamente, representado numa grelha, que inicialmente está completamente coberta. O objetivo é destapar a grelha sem rebentar nenhuma bomba. Nesta adaptação será considerada uma matriz que representará o campo minado, composta pelos números inteiros -1 e 0. O número -1 representará uma bomba, e o 0 a ausência de bomba.
Para apoiar construção deste jogo deverá implementar um módulo `mines.py`, com as seguintes funções:

`ler_jogo(ficheiro)` Lê um campo de bombas de um ficheiro de texto, e retorna a sua representação em matriz, i.e., lista de listas. Está disponível o ficheiro `minefield.csv` como exemplo.

`contar_bombas(m)` Recebe como parâmetro a matriz que representa o campo minado e retorna o número de bombas (-1) presentes na matriz.

`contar_bombas_adjacentes(m, x, y)` Recebe como parâmetro a matriz que representa o campo minado e retorna o número de bombas (-1) adjacentes à posição x, y. O número de bombas adjacentes não inclui a posição x,y. A adjacência é determinada por uma vizinhança de 8 posições na grelha, à volta da posição indicada.

`bomba_mais_proxima(m, x y)` Recebe como parâmetro a matriz que representa o campo minado e uma posição (x, y), e retorna dois valores com as coordenadas da bomba mais próxima da posição indicada.

Parte 3 (4 valores)

Pretende-se um programa que converta imagens no formato BMP a cores para monocromático, i.e., preto e branco.

Uma imagem BMP é uma grelha de pixeis. Neste programa, um pixel representa um ponto da imagem, com informação de intensidade para 3 cores: vermelho (R), verde (G), e azul (B). O valor RGB de um pixel com cor preta é (0,0,0), e o valor de um pixel com cor branca é (255,255,255).

Pode consultar mais informação sobre o formato BMP em `https://en.wikipedia.org/wiki/BMP_file_format`.

O módulo `bmplib.py` tem duas funções implementadas:

`ler_imagem(ficheiro)` Retorna uma lista de tuplos, onde cada tuplo representa um pixel, i.e., três valores inteiros entre 0 e 255.

`escrever_imagem(ficheiro, ficheiro_mono, imagem_mono)` Escreve uma imagem BMP com a informação da lista de tuplos `imagem_mono` em `ficheiro_mono`, de acordo com o formato da imagem BMP original em `ficheiro`.

O módulo `bmplib.py` depende de uma biblioteca python: a `Pillow`. Instale no seu sistema, na linha de comandos, com `conda install Pillow`, ou `pip install Pillow`.
Implemente um módulo `bmpmono.py` com a seguinte funcionalidade:

ponto de entrada Consulta dois parâmetro da linha de comandos: um para o nome (ou caminho) do ficheiro com a imagem original, e outro com o nome (ou caminho) do ficheiro onde a versão monocromática será gravada.

Deve informar o utilizador se o número de parâmetros utilizado for errado.

Invoca a função `converter_imagem` para efetuar a conversão.

`converter_imagem(ficheiro, ficheiro_mono)` Lê a imagem de `ficheiro`, converte os pixeis para monocromático, e escreve o resultado em `ficheiro_mono`.

Exemplo de utilização:
`python bmpmono.py imagem.bmp imagem_mono.bmp`

Parte 4 (4 valores)

Considere o ficheiro `stock.csv` com informação sobre o *stock* de uma frutaria.

Um cesto de compras deve ser representado por uma lista de dicionários com a seguinte estrutura:

```
{
    'produtos': [
        {
            'nome': str,
            'quantidate': str,
            'preco_unitario': float
        }
    ],
    'total': float
}
```

A chave `total` representa o valor total dos produtos no cesto.
Implemente um módulo `frutaria.py` com as seguintes funções:

`gerar_cesto(stock, min_produtos, max_produtos)` Gera um cesto de compras com uma quantidade aleatória tipos de fruta, entre `min_produtos` e `max_produtos`.

Cada tipo de fruta ter uma quantidade entre `min_produtos` e `max_produtos`.

`atualizar_stock(stock, cestos)` Atualiza a lista `stock`, removendo as quantidades de produtos na lista de cestos de compras `cestos`.

Caso algum produto chegue à quantidade zero, deve ser removo de `stock`.

Parte 5 (4 valores)

Construa um programa que simula um sistema de aquisição de passes do Metropolitano de Lisboa. Considere a seguinte tabela de preços:

Passes	Validade espacial	Validade temporal	Normal	Social B	Social A	Sub 23
Metropolitano	AML	mensal	40.0	30.0	20.0	16.0
Municipal	Lisboa ou Amadora ou Odivelas	mensal	30.0	22.5	15.0	12.0
-12	AML	gratuito	0.0	0.0	0.0	0.0
+16	AML	mensal	20.0	20.0	20.0	20.0

Implemente o programa num pacote `passes` com, pelo menos, um módulo `programa.py`. As instruções a suportar são:

`R Nome NIF Idade Escalão Saldo_Diponivel` Registo dos clientes, indicando o nome, nif, idade, escalão do passe (*Normal*, *Social B*, *Social A*, *Sub23*) e o saldo disponível para compras.

`EP NIF TipoPasse` Escolha do tipo de passe a adquirir. O cliente poderá introduzir um dos seguintes tipos: *Metropolitano*, *Municipal Lisboa*, *Municipal Amadora*, *Municipal Odivelas*, *-12* ou *+65*.

1. Se o cliente reunir condições para o tipo de passo indicado, e possuir saldo suficiente para realizar a compra é retornado, na mesma linha, separados por espaços, o preço, a validade espacial, e a data de validade do passe.

   Caso contrário, é indicado:

   `Saldo insuficiente.`

2. O cliente só pode adquirir o passe *-12* se a sua idade for igual ou inferior a 12 anos, caso contrário é apresentado:

   `Não apresenta os requisitos necessários para adquirir este passe.`

3. O cliente só pode adquirir o passe *+65* se a sua idade for igual ou superior a 65 anos, caso contrário é apresentado:

   `Não apresenta os requisitos necessários para adquirir este passe.`

`CV NIF` Consulta da validade do passe. É retornado o tipo de passe e a respetiva validade.Caso o passe se encontre fora de validade é apresentado:

`Data de validade expirada.`

`G Ficheiro` Grava a informação atual no ficheiro. Toda a informação deve ficar registada.

`L Ficheiro` Lê a informação de um ficheiro.