Enunciados de questões de P1 – 2023/1

Questão 1 (total: 4.0 pontos) Resolva toda essa questão no arquivo disponibilizado *templateDaProvaDicionario.py*. Respeite os espaços sinalizados para escrever cada função pedida. Há dicionários exemplos e testes fornecidos.

Questão 1a (total: 1.5 pontos) Dicionário do Imposto de Renda
Considere um dicionário de pagamentos por CPF de pagadores (um dicionário de dicionários) em que cada elemento/item é:

CHAVE: CPF do pagador (CPFpagador)
VALOR: dicionário com CPFs dos recebedores e respectivos valores pagos

Escreva uma função, denominada **exibeTotalRecebidoPorUmCPF**, que:
- receba um dicionário com a descrição acima e um CPF (cpf);
- exiba o valor total que esse CPF recebeu considerando as informações dos CPFs pagadores

Para o dicionário exemplo disponibilizado na área de teste da questão 1A e o CPF '67167776-73' deve ser exibido:
67167776-73 recebeu no total R$13800.00

Questão 1b (total: 2.5 pontos): Dicionário dos tipos (fontes) de energia.
Considere um dicionário como o **dic_energia** disponibilizado na área de testes. É um dicionário de dicionários com as informações sobre os diferentes tipos de energia.
Obs: as informações não são confiáveis (obtidas com a ajuda do chatgpt)

O dicionário externo tem:
CHAVE: tipo de energia
VALOR: dicionário interno com informações importantes sobre o tipo (fonte) de energia
E obrigatoriamente as CHAVEs de todos os dicionários internos são:
tecnologia => tecnologia ou método usado para gerar energia a partir dessa fonte,
capacidade => capacidade máxima ou potencial de geração desse tipo de energia, expressa em kW,
geracao => atual quantidade de energia gerada desse tipo, expressa em kW,
eficiencia => quão eficiente é a conversão dessa fonte de energia em energia utilizável, expressa como percentagem entre 0 e 1
custo_por_kw => custo para gerar um kW desse tipo de energia (em uma determinada moeda)
vantagens => lista de vantagens ou benefícios desse tipo de energia

Exemplo de um elemento do **dic_energia**:
"solar": {"tecnologia": "Fotovoltaica", "capacidade": 1000, "geracao": 800, "eficiencia": 0.8, "custo_por_kw": 0.12, "vantagens": ["renovavel", "limpa", "baixos custos operacionais"]}

Escreva uma função, denominada **cria_dic_por_vantagem**, que:
- receba um dicionário por energia como o descrito
- construa e retorne um dicionário por Vantagem (ou benefício), em que cada item é:
CHAVE: a vantagem (benefício)
VALOR: lista dos tipos de energia que apresentam essa vantagem.

Por exemplo, para o dicionário **dic_energia** fornecido na área de testes, a função retornaria o seguinte dicionário:
{'renovavel': ['solar', 'eolica', 'hidreletrica', 'geotermica', 'biomassa'], 'limpa': ['solar'], 'baixos custos operacionais': ['solar'], 'sem emissões': ['eolica'], 'recurso abundante': ['eolica'], 'baixas emissoes': ['hidreletrica', 'geotermica', 'nuclear'], 'longa vida util': ['hidreletrica'], 'confiavel': ['geotermica', 'nuclear'], 'alta densidade energetica': ['nuclear'], 'utilizacao de residuos': ['biomassa']}

Questão 2 (total: 3.0 pontos): Resolva toda essa questão no arquivo disponibilizado templateDaProvaOO.py. Respeite os espaços sinalizados para escrever cada classe. Há testes fornecidos para serem executados. O resultado esperado é apresentado logo após o enunciado.

Demonstre que compreendeu bem as relações entre classes considerando a classe ***Data*** dada e criando duas novas classes como pedido nas questões **2a)** e **2b)**. A classe ***Data*** **que não pode ser alterada** e está disponibilizada no arquivo dataP1.py, tem a seguinte descrição:

construtor (__init__)	Recebe dia, mês e ano (inteiros) e cria um objeto. Caso dia, mês e ano não sejam fornecidos, cria um objeto com a data atual (do sistema), portanto, hoje (25/04/2023), **Data() cria um objeto data usando a data atual 25/04/2023.**
apresentação (__str__ e __repr__)	Retorna uma string com os valores dos atributos no formato dd/mm/aaaa
>	Retorna True se a data da esquerda é mais recente que a data da direita. False, caso contrário.
<	Retorna True se a data da esquerda é mais antiga que a data da direita. False, caso contrário.
==	Retorna True se a data da esquerda é a mesma da direita. False, caso contrário
-	Recebe uma quantidade de dias(x) e retorna a data de x dias anteriores

2.a) (1.5 pontos) Escreva a classe **Arquivo** para representar um arquivo texto. A classe **Arquivo** usa obrigatoriamente a classe **Data**.
ATENÇÃO: **TODOS** os métodos da classe **Arquivo** usam direta ou indiretamente métodos da classe **Data.**

Um *arquivo* tem os seguintes atributos:

- nome
- autor
- data da criação
- texto
- data da última modificação

Disponibilize os seguintes métodos:

- construtor (__init__): recebe obrigatoriamente o nome do arquivo, o autor e a data de criação (objeto Data). Se não especificado, o texto é vazio. A data da última modificação é sempre a data da criação.
- apresentação (__str__ e __repr__): retorna uma string com: o nome do arquivo seguido de **'.txt'**, o autor, a data de criação e o tamanho do arquivo (método ***tamanho***)
- + (__add__): recebe outro objeto da classe arquivo e retorna um novo arquivo cujo nome é a concatenação dos nomes dos arquivos, o autor é 'sistema', o texto é texto do arquivo da esquerda, seguido de '\n' seguido do texto do arquivo da direita e a data da criação é a data atual.

- Um arquivo pode:
  - calcular e retornar seu tamanho. O tamanho do arquivo é o comprimento do texto (método: ***tamanho***)
  - substituir seu texto. Recebe um texto e um objeto Data como parâmetros. Este método deve substituir o texto do arquivo para o recebido e a data de atualização do arquivo é alterada para a data recebida. (método: **substituiTexto**)
  - adicionar texto ao final do já existente. Recebe como parâmetros um texto e um objeto Data. Inclui o texto recebido no final do texto do arquivo e a data de atualização do arquivo é alterada para a data recebida. (método: ***adicionaTexto***)
  - exibir texto do arquivo. (método: ***exibeTexto***)
  - saber se o arquivo foi alterado em uma determinada data: recebe como parâmetro uma data e retorna True se a última alteração do arquivo foi nesta data ou False, caso contrário (método: **ultimaAlteracaoNaData)**

2.b) (1.5 pontos) Escreva a classe **Pasta**.

Uma pasta quando criada tem:

- o nome da pasta
- uma lista de arquivos (inicialmente vazia).

Disponibilize os seguintes métodos:

- construtor (__init__): recebe obrigatoriamente o nome da pasta. A lista de arquivos é criada vazia
- apresentação (__str__ e __repr__): retorna uma string com o nome da pasta e o número de arquivos contidos na pasta
- Uma pasta pode:
  - incluir um novo arquivo em sua lista, recebendo para isso um arquivo **criado** (método: ***incluiArquivo***)
  - exibir os dados de todos os seus arquivos. Caso a pasta não tenha arquivos, escrever a mensagem "Pasta Vazia (método: ***exibeArquivos***)
  - mostrar os arquivos que foram alterados em uma data recebida como parâmetro. (método: **alteradosNaData).**

Teste as classes construídas por você, executando o código que se encontra comentado na área de testes no arquivo 'templateDaProvaOO.py', na seção "# Área de teste da Questao 2:". *A saída esperada para o teste é:*

```
==================================
ARQUIVOS CRIADOS
==================================
Arquivo: comprasFrutas.txt - Autor: fifi.
Criado em 12/04/2023 com 26 bytes
Texto do arquivo:
abacate,pera,abacaxi,manga

Arquivo: comprasFrutas.txt - Autor: fifi.
Criado em 12/04/2023 com 41 bytes
Texto do arquivo:
abacate,pera,abacaxi,manga,banana,laranja

-->A última alteração do arquivo FOI em 19/04/2023
--------------------------------
--------------------------------
Arquivo: comprasBebidas.txt - Autor: guga.
Criado em 10/04/2023 com 0 bytes
Texto do arquivo:


Arquivo: comprasBebidas.txt - Autor: guga.
Criado em 10/04/2023 com 9 bytes
Texto do arquivo:
suco,água
--------------------------------
--------------------------------
Arquivo: comprasFrutascomprasBebidas.txt - Autor:
sistema.
Criado em 25/04/2023 com 51 bytes
Texto do arquivo:
abacate,pera,abacaxi,manga,banana,laranja
suco,água


-->A última alteração do arquivo FOI em 25/04/2023
--------------------------------
--------------------------------
Arquivo: convBibi.txt - Autor: bibi.
Criado em 01/04/2023 com 39 bytes
Texto do arquivo:
juca,keko,kaka,lilo,mano,mimi,dora,zeze
--------------------------------
--------------------------------
==================================
PASTA CRIADA
==================================
PASTA: festaNiver  Qtdade de arquivos: 0
--------------------------------
Arquivos da pasta
PASTA VAZIA

==================================
ARQUIVOS INCLUIDOS NAS PASTAS
==================================
PASTA: festaNiver  Qtdade de arquivos: 4
Arquivos da pasta
Arquivo: comprasFrutas.txt - Autor: fifi.
Criado em 12/04/2023 com 41 bytes
Arquivo: comprasBebidas.txt - Autor: guga.
Criado em 10/04/2023 com 9 bytes
Arquivo: comprasFrutascomprasBebidas.txt - Autor:
sistema.
Criado em 21/04/2023 com 51 bytes
Arquivo: convBibi.txt - Autor: bibi.
Criado em 01/04/2023 com 39 bytes

--------------------------------

Arquivos alterados hoje na pasta PASTA: festaNiver
Qtdade de arquivos: 4

--> Arquivo: comprasFrutascomprasBebidas.txt -
Autor: sistema.
Criado em 25/04/2023 com 51 bytes
```