Fundação CECIERJ - Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AP2 1° semestre de 2019

IMPORTANTE

- Serão aceitos apenas soluções escritas na linguagem Python 3.
- Prova sem consulta e sem uso de qualquer aparato eletrônico.
- Use caneta para preencher o seu nome e assinar nas folhas de questões e de respostas.
- Você pode usar lápis para responder as questões.
- Ao final da prova, devolva as folhas de questões e as de respostas.
- Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

1ª Questão (2,5 pontos)

Faça um programa que leia da entrada padrão o nome de um arquivo texto, contendo várias palavras por linha, e produza um conjunto (`set`) de todas as palavras do arquivo que possuam pelo menos dois caracteres repetidos. Ao final, mostre o conjunto produzido, formatando conforme apresentado nos exemplos.

Exemplos

Entrada	
Entrada Padrão	Arquivo `teste1.txt`
`teste1.txt`	*<arquivo em branco>*
Saída Padrão	
`Nenhuma palavra com caracteres repetidos foi encontrada!!!`	

Entrada	
Entrada Padrão	Arquivo `teste2.txt`
`teste2.txt`	`1+ +1 -1` `ama 1- +1- corre -1.1+ +1+`
Saída Padrão	
`Palavras com caracteres repetidos = {'+1+', 'ama', '-1.1+', 'corre'}`	

Entrada	
Entrada Padrão	Arquivo `carta.txt`

`carta.txt`	`o tempo perguntou pro tempo quanto tempo o tempo tinha`
	`o tempo respondeu pro tempo que o tempo tinha o mesmo tempo`
	`que o tempo tem mora na filosofia`

Saída Padrão	
`Palavras com caracteres repetidos = {'respondeu', 'filosofia', 'mesmo', 'perguntou'}`	

Distribuição de Pontos

Entrada – 0,3 pontos; Processamento – 2,0 pontos; Saída – 0,2 pontos.

2ª Questão (2,5 pontos)

Utilizando subprogramação, faça um programa que produza um dicionário (`dict`) de todas as palavras (chaves) com suas respectivas contagens (valores) de ocorrência. Processe o arquivo e gere o dicionário solicitado. Mostre o dicionário produzido, ordenado pela chave, listando a chave e seu respectivo valor, um por linha. Remova do dicionário todos os pares (chave, valor) que tenham chave de comprimento ímpar e valor par. Ao final, mostre, ordenado pela chave, o conteúdo do dicionário de ocorrências. Suponha que cada linha do arquivo possua várias palavras. Formate as saídas conforme apresentado nos exemplos.

Sugestão

Utilize a operação `pop` para remover um par `(chave, valor)` do dicionário `meuDict`:

```
meuDict.pop(chave)
```

Mas fique atento, no momento da remoção o dicionário não poderá estar sendo iterado.

Exemplos

Entrada	
Entrada Padrão	Arquivo `teste1.txt`
`teste1.txt`	*<arquivo em branco>*
Saída Padrão	
`O dicionário ficou vazio!!!`	

Entrada	
Entrada Padrão	Arquivo `carta.txt`

`carta.txt`	`o tempo perguntou pro tempo quanto tempo o tempo tinha`
	`o tempo respondeu pro tempo que o tempo tinha o mesmo tempo`
	`que o tempo tem mora na filosofia`

Saída Padrão	

`Dicionário ordenado pelas palavras:`
`  o ocorreu 1 vez`
`  filosofia ocorreu 1 vez`
`  mesmo ocorreu 1 vez`
`  mora ocorreu 1 vez`	

```
    na ocorreu 1 vez
    o ocorreu 5 vezes
    perguntou ocorreu 1 vez
    pro ocorreu 2 vezes
    quanto ocorreu 1 vez
    que ocorreu 2 vezes
    respondeu ocorreu 1 vez
    tem ocorreu 1 vez
    tempo ocorreu 9 vezes
    tinha ocorreu 2 vezes
Dicionário, após remover chaves de comprimento par, ordenado pelas palavras:
    o ocorreu 1 vez
    filosofia ocorreu 1 vez
    mesmo ocorreu 1 vez
    o ocorreu 5 vezes
    perguntou ocorreu 1 vez
    pro ocorreu 2 vezes
    que ocorreu 2 vezes
    respondeu ocorreu 1 vez
    tem ocorreu 1 vez
    tempo ocorreu 9 vezes
    tinha ocorreu 2 vezes
```

Distribuição de Pontos

Entrada (com conversão de tipos) – 0,5 pontos; Gerar dicionário inicial – 0,5 pontos; Filtrar dicionário – 1,0 ponto; Formatação e impressão da saída – 0,5 pontos.

3ª Questões (5,0 pontos)

Essa questão é dividida em três partes. Cada parte será avaliada individualmente, pois a implementação de uma não implica na capacidade de implementação de outra.

Primeira Parte (1,5 pontos)

Implemente UMA das duas funções descritas nessa parte. Escolha a função que você achar melhor, pois a codificação é equivalente. Cada uma das funções é especializada na leitura de um arquivo binário diferente: o arquivo de alunos e o arquivo de disciplinas. Essas funções recebem um único argumento, o nome do arquivo em questão (`str`), e retornam, cada uma, um dicionário (`dict`) onde o campo a ser utilizado como chave é indicado abaixo e o valor associado à chave deverá ser uma tupla contendo todos outros campos contidos no arquivo.

O arquivo de alunos armazena os dados conforme a seguinte estrutura:

Matrícula: texto com exatos 9 caracteres (essa é a chave a ser utilizada no dicionário).
CPF: texto com exatos 14 caracteres.
Nome: texto com até 50 caracteres (caracteres não usados devem ser descartados na leitura).

O arquivo de disciplinas armazena os dados conforme a seguinte estrutura:

Código: texto com exatos 10 caracteres (essa é a chave a ser utilizada no dicionário).
Nome: texto com até 50 caracteres (caracteres não usados devem ser descartados na leitura).
Créditos: valor inteiro com 4 bytes.

Observe que essa parte da questão não requer a implementação do programa principal nem de comunicação com o usuário.

Distribuição de Pontos: Definição das funções – 0,2 pontos; Leitura de arquivos binários e conversão de tipos – 1,3 pontos.

Segunda Parte (2,0 pontos)

Utilizando subprogramação, implemente um procedimento. Esse procedimento recebe como argumentos o nome de um arquivo binário (`str`), a matrícula de um aluno (`str`), o código de uma disciplina (`str`) e a média final obtida pelo aluno nessa disciplina (`float`). A função deve abrir o arquivo binário ou cria-lo, caso ele não exista. Uma vez aberto o arquivo, o procedimento deve procurar por um registro com mesma matrícula e código. Caso o encontre, a média escrita nesse registro deve ser substituída. Caso não encontre, um novo registro deve ser adicionado no final deste arquivo.

O arquivo de médias armazena os dados conforme a seguinte estrutura:

Matrícula: texto com exatos 9 caracteres.
Código: texto com exatos 10 caracteres.
Média: valor em ponto flutuante com 4 bytes.

Observe que essa parte da questão não requer a implementação do programa principal nem de comunicação com o usuário.

Distribuição de Pontos: Definição do procedimento – 0,2 pontos; Busca pelo registro – 1,0 ponto; Escrita de arquivo binário – 0,8 pontos.

Terceira Parte (1,5 ponto)

Escreva um programa que solicite que o usuário informe o nome do arquivo contendo os registros de alunos e o nome dos registros de disciplinas. Em seguida, seu programa deve chamar as funções implementadas na primeira parte da questão para ler o conteúdo desses arquivos e retornar dicionários (você só precisou implementar uma, mas chame a outra também como se ela tivesse sido implementada). Por fim, o usuário deve informar uma sequência de matrículas de alunos, códigos de disciplina e médias finais para que essas sejam armazenadas no arquivo "medias.dat" utilizando o procedimento implementado na segunda parte desta questão. Antes de chamar o procedimento de escrita, seu programa deve verificar se a matrícula e o código de disciplina são válidos. São considerados válidos matrículas e códigos existentes nos dicionários de, respectivamente, alunos e disciplinas. Dados inválidos devem ser ignorados e não podem ser escritos no arquivo de médias. O usuário indicará o texto "fim" na matrícula do aluno para informar que não existem mais registros a serem armazenados.

Distribuição de Pontos: Leitura dos dados via entrada padrão – 0,2 pontos; Verificação de validade dos dados – 0,5 pontos; Estrutura do programa principal – 0,8 pontos.

Dica: Os tamanhos (quantidade de bytes) assumidos para formatos nativos de valores inteiros e de valores em ponto flutuante lidos ou escritos de arquivos binários podem variar de plataforma para plataforma. Ou seja, podem ocorrer problemas de compatibilidade entre programas que rodam perfeitamente em computadores que assumem determinados tamanhos para tipos primitivos, mas que não rodam corretamente em computadores que assumem outros tamanhos para o mesmo tipo. Para forçar a leitura e escrita assumindo os tamanhos padrão (*standard*) que são indicados na Aula 12 e ficar livre de problemas de compatibilidade, inclua o símbolo "=" na frente do formato indicado nas funções `.pack` e `.unpack` de `struct`. Por exemplo, `struct.unpack('i', bloco)` converte o bloco de bytes em um valor inteiro, mas o tamanho do bloco é dependente da plataforma (não é necessariamente de 4 bytes), enquanto que `struct.unpack('=i', bloco)` converte blocos de 4 bytes em valores inteiros, independentemente da plataforma.

Boa Avaliação!