Fundação CECIERJ - Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AP3X 1° semestre de 2020

IMPORTANTE

- As respostas (programas) deverão ser entregues pela plataforma em um arquivo ZIP contendo todos os arquivos de código fonte (extensão ".py") necessários para que os programas sejam testados. Respostas entregues fora do formato especificado, por exemplo, em arquivos com extensão ".pdf", ".doc" ou outras, não serão corrigidas.
- Serão aceitos apenas soluções escritas na linguagem Python 3. Programas com erro de intepretação não serão corrigidos. Evite problemas utilizando tanto a versão da linguagem de programação (Python 3.X) quanto a IDE (PyCharm) indicadas na Aula 1.
- Quando o enunciado de uma questão inclui especificação de formato de entrada e saída, tal especificação deve ser seguida à risca pelo programa entregue. Atender ao enunciado faz parte da avaliação e da composição da nota final.
- Os exemplos fornecidos nos enunciados das questões correspondem a casos específicos apontados para fins de ilustração e não correspondem ao universo completo de entradas possíveis especificado no enunciado. Os programas entregues devem ser elaborados considerando qualquer caso que siga a especificação e não apenas os exemplos dados. Essa é a prática adotada tanto na elaboração das listas exercícios desta disciplina quanto no mercado de trabalho.
- Faça uso de boas práticas de programação, em especial, na escolha de identificadores de variáveis, subprogramas e comentários no código.
- As respostas deverão ser entregues pela atividade "Entrega de AP3X" antes da data final de entrega estabelecida. Não serão aceitas entregas tardias ou substituição de respostas após término do prazo.
- As APXs são um mecanismo de avaliação individual. As soluções podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual. Respostas plagiadas não serão corrigidas.

1ª Questão (2,0 pontos)

Faça um programa, contendo subprogramas, que leia o nome de um arquivo texto contendo várias palavras por linha. Produza um dicionário com todas as palavras do arquivo e suas respectivas contagens de ocorrência. Escreva, ordenado alfabeticamente todas as palavras que ocorreram mais que cinco vezes no arquivo.

Exemplo

Entrada Padrão	Saída Padrão
carta.txt	o tempo
Conteúdo do Arquivo "carta.txt"	
o tempo perguntou pro tempo quanto tempo o tempo tinha o tempo respondeu para o tempo que o tempo tinha o mesmo tempo que o tempo tem	

Distribuição de Pontos

Entrada – 0,3 pontos; Processamento – 1,5 ponto; Saída – 0,2 pontos.

2ª Questão (3,0 pontos)

Faça um programa, contendo subprogramas, que leia da entrada padrão o nome de um arquivo texto, contendo matriz bidimensional de números inteiros. Suponha que cada linha do arquivo contenha todos os números de uma linha da matriz, separados por espaços em branco. Leia o arquivo e escreva na saída padrão qual(is) é(são) a(s) linha(s) que possuem apenas números primos. Escreva também qual(is) é(são) a(s) linha(s) que possuem apenas números primos.

Definição

Um número inteiro é primo se e somente se for maior que um e for apenas divisível por ele e por um.

Exemplos

Entrada Padrão
matNumeros.txt
Conteúdo do Arquivo "matNumeros.txt"
5 2 1 3 7 3 7 2 7 5 4 3 3 2 4
Saída Padrão
Relação de linha(s) e coluna(s) com todos elementos primos: Linha 2 Coluna 2 Coluna 4

Distribuição de Pontos

Entrada – 0,5 pontos; Processamento – 3,0 pontos; Saída – 0,5 pontos.

3ª Questão (5,0 pontos)

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

Primeira Parte (2,0 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 produtos e o arquivo de compras. 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 produtos armazena os dados conforme a seguinte estrutura:

Código de Barras: texto com exatos 13 caracteres em codificação UTF-8 (essa é a chave a ser utilizada no dicionário).
Nome do Produto: texto com até 50 caracteres em codificação UTF-8 (caracteres não usados devem ser descartados na leitura).
Preço: valor em ponto flutuante com 4 bytes.

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

Código da Compra: texto com exatos 10 caracteres em codificação UTF-8 (essa é a chave a ser utilizada no dicionário).
Nome do Comprador: texto com até 256 caracteres em codificação UTF-8 (caracteres não usados devem ser descartados na leitura).

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 – 1,8 pontos.

Segunda Parte (3,0 pontos)

Utilizando subprogramação, implemente um procedimento. Esse procedimento recebe como argumentos o nome de um arquivo binário (`str`), o código de barras de um produto (`str`), o código de uma compra (`str`) e a quantidade de produtos comprados (`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 mesmo código de compra e código de barras. Caso o encontre, a quantidade comprada escrita nesse registro deve ser substituída. Caso não encontre, um novo registro deve ser adicionado no final deste arquivo com as informações fornecidas.

O arquivo de preenchimento das compras armazena os dados conforme a seguinte estrutura:

Código da Compra: texto com exatos 10 caracteres em codificação UTF-8.
Código de Barras: texto com exatos 13 caracteres em codificação UTF-8.
Quantidade Comprada: valor inteiro com 4 bytes.

Observe que essa parte da questão também 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 – 1,5 ponto; Escrita de arquivo binário – 1,3 pontos.

Dicas

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!