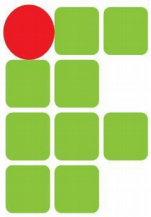

Instituto Federal do Espírito Santo
Campus Serra
Coordenadoria de Informática
Curso Superior Bacharelado em Sistemas de Informação

2ª Avaliação de Programação 2 – Semestre 2018-1

Turma Prof. Ernani Filho

Leia com atenção

- **O material para a prova estará contido na pasta *paraProva1.zip*.**
- **Para a entrega da prova, compacte os arquivos.py em um arquivo .ZIP e faça a entrega como uma atividade no moodle. Nomeie o arquivo zip com *<seu nome>.zip*.**
- **Toda questão estará correta somente se produzir os resultados solicitados e obedecer às especificações e restrições fornecidas no enunciado.**
- **As soluções das questões devem ser construídas utilizando APENAS o conteúdo de python visto em sala de aula.**
- **EVITE PERDER PONTOS POR FALTA DE INFORMAÇÃO: na dúvida sobre quais recursos/comandos da linguagem usar, solicite a orientação do professor.**

Instruções e restrições para TODAS as questões da prova

(a) A escolha dos nomes de variáveis e parâmetros de entrada das funções também FAZ parte do conteúdo avaliado.
(b) NÃO utilizar interrupções de loop do tipo *break* nem resumos do tipo *continue*.
(c) Quando for necessário algum tipo de sequência, fazer uso APENAS de listas, tuplas, strings e dicionários.
(d) Uma vez definido o tipo da variável, NÃO alterá-lo ao longo de todo o programa.
(e) Para responder às questões, utilizar EXATAMENTE os nomes das funções fornecidos nos enunciados.
(f) Nos arquivos q1.py e q2.py, escreva o SEU NOME nos cabeçalhos dos arquivos.

Observação

O único equipamento eletrônico permitido durante a confecção da prova é o computador do laboratório. A prova será anulada caso esta regra seja violada.

Questão 1 (30 pontos, tempo estimado: casa)

Palavras-chave: arquivo, ordenação, particionamento, intercalação.

No arquivo **main2aAVAL.py** fornecido com o material da prova, construa o código das funções pedidas nos itens a), b) e c) a seguir:

Dados de entrada:

[1] ***Planilhas_Enem_2015_download.csv***: arquivo texto contendo uma base de dados csv organizada da seguinte forma, <nome, endereço, e-mail, telefone>. O separador de campos é o ponto e vírgula.

a) Função ***particiona(<str nome arquivo csv>, <quantidade de partes: int>): Lista Strings***. A função deve dividir o conteúdo do arquivo csv, cujo nome se encontra no primeiro parâmetro da função, em uma quantidade de pequenos arquivos. Este arquivo deve ser processado LINHA a LINHA pois será considerado que ele é muito grande para caber todo em memória. A quantidade de arquivos partes está definida no segundo parâmetro da função. A divisão deve ser feita de tal forma que os arquivos partes, também csv, tenham entre si quantidades semelhantes de números de linhas, e linhas inteiras. A função retorna como saída uma lista com os nomes dos arquivos partes. Os nomes dos arquivos partes devem ser da forma: *<nome arquivo principal>00x.csv*, onde x é um inteiro com o número da fração do arquivo. O primeiro arquivo é p de número (x) igual a 1. Maiores detalhes foram discutidos em aula e a solução deve incluir esses detalhes no código da resposta. **(10 pontos)**

Observação: pesquise qual, ou quais, recursos da linguagem python podem ser utilizados para se calcular e descobrir o tamanho de um arquivo em disco SEM que seja necessário ler todo o conteúdo do arquivo. Lembre-se da discussão realizada em aula.

b) Função ***intercala(<Lista nomes de arquivos partes>, <String nome arquivo completo>: String nome arquivo***. A função recebe como entrada uma lista de nomes de arquivos csv, todos ordenados segundo um determinado critério. A função deve processar esses arquivos LINHA A LINHA e gerar um único arquivo completo ordenado. A função deve retornar o nome do arquivo total. O nome do arquivo total, completo, é passado no segundo parâmetro da função. **(10 pontos)**

c) Função ***ordena_quick(<Lista dados csv>, <int inicio>, <fim>): <Lista dados csv>***. Pesquise um método de ordenação chamado quicksort. O método é bem conhecido na área de computação. Recorra ao professor em caso de dúvidas. Construa a função de forma que ela receba uma lista de listas. As listas mais internas têm como elementos os dados extraídos das linhas de um arquivo csv. Ver arquivo modelo1.py para mais detalhes.

Assim, cada lista interna representa uma linha de um arquivo csv cujos os dados já foram separados e transformados na lista.

Chaves de classificação:

- ordenar o arquivo por ordem crescente de:

- 1ª chave, coluna 2, SIGLA DA UF (Estado da federação);
- 2ª chave, coluna 3, NOME MUNICÍPIO;
- 3ª chave, coluna 1, NOME DA ENTIDADE.

Processo de correção:

O trabalho será corrigido segundo o algoritmo a seguir:

Início

Passo 1: montar matriz de similaridade para verificação de plágio.

Passo 2:

Para cada aluno **faça**:

if NÃO houve detecção de plágio[*] **então**:

if arqs de saída da aplic. principal (main2aAVAL.py) estiverem ordenados **então**:

- Corrigir as funções: *particiona(..)*, *ordena_ quick(..)* e *intercala(..)*; [**]

else:

- corrigir as funções: *particiona(..)*, *ordena_ quick(..)* e *intercala(..)*; [**]
- Aplicar desconto de 50% na nota final;

eles:

- Chamada dos alunos para esclarecimentos;
- Nota avaliação igual a zero;

fim para

Passo 3: chamada dos alunos para explicação rápida do código enviado.

Fim.

[*] Detecção de plágio

A detecção de plágio será avaliada por meio do cálculo de similaridade entre códigos. Os seguintes programas serão utilizados para avaliar a similaridade:

- MOSS (Measure of Software Similarity) system provided by Stanford: https://theory.stanford.edu/~aiken/moss/
- Jplag (detecting software plagiarism) system provided by Institute for Program Structures and Data Organization: http://jplag.ipd.kit.edu/

[**] Correção das funções *particiona(..)*, *ordena_quick(..)* e *intercala(..)*

Cada função receberá um valor de 0 a 5: **0** (função não faz o que é pedido no enunciado); **1** (função não trata os parâmetros de entrada corretamente, nem retorna o tipo do dado que foi pedido); **2** (função trata os parâmetros e tipo de saída corretamente mas o código não obedece à especificação); **3** (função trata os parâmetros e tipo de saída corretamente, o código obedece à especificação mas não resolve o problema); **4** (função trata os parâmetros e tipo de saída corretamente , o código obedece à especificação mas não resolve totalmente o problema); **5** (função trata os parâmetros e tipo de saída corretamente , o código obedece à especificação e resolve totalmente o problema).

Conteúdo do arquivo *aval-2-2018-1.zip*

- *prova2-BS2-2018-1-para-casa.pdf*: este documento pdf que você está lendo.
- *main2aAVAL.py*: aplicação principal que executará os códigos das funções e ordenar o arquivo csv massivo.
- *modelo1.py*: exemplo da estrutura de dados para o qual devem ser convertidos os arquivos partes lidos da memória e submetidos ao quick_sort.
- *Planilhas_Enem_2015_download.csv*: arquivo csv de dados de entrada contendo os dados a serem classificados via o processo de intercalação descrito no enunciado deste documento. O cabeçalho com os títulos das colunas NÃO deve ser processado
- *Planilhas_Enem_2015_download_sorted.csv*: resposta da atividade. Arquivo csv massivo do item anterior devidamente classificado pelas chaves de classificação pedidas. Para facilitar a conferência, as colunas classificadas foram deslocadas para as posições iniciais.

ෆ ෆ

Bom trabalho!