Fundação CECIERJ - **Vice-Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AD2 – 1° semestre de 2018

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.
- 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 AD2" antes da data final de entrega estabelecida no calendário de entrega de ADs. Não serão aceitas entregas tardias ou substituição de respostas após término do prazo.
- As ADs 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.
- 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.

1ª Questão (2,0 pontos)

A classificação de candidatos em um concurso é definida em função da média ponderada de suas notas nas cinco avaliações realizadas. Em caso de empate nas médias, os elaboradores do concurso utilizam a idade dos candidatos empatados como critério de desempate assumindo que candidatos mais velhos têm preferência sobre candidatos mais jovens. Caso o empate persista, o critério de desempate adotado é a ordem alfabética dos nomes.

Você foi contratado para escrever um programa que monte e exiba a classificação final dos candidatos desse concurso seguindo o critério de ordenação definido no parágrafo anterior. Os dados são informados para seu programa via entrada padrão e seu programa deve emitir a ordem de classificação dos candidatos na saída padrão.

Restrição

Rotinas de ordenação prontas e disponíveis na API do Python ou em bibliotecas externas à API não poderão ser utilizadas na solução desta questão. Logo, seu programa deverá implementar um dos métodos de ordenação vistos na Aula 7.

Entrada

A primeira linha da entrada contém cinco valores em ponto flutuante representando o peso das avaliações. A soma dos pesos é igual a 10,0 (dez). Os valores são separados por espaços em branco. A segunda linha da entrada indica a quantidade N de candidatos. Cada uma das N linhas seguintes contém os dados de um dos candidatos. A composição de cada uma dessas linhas é dada por uma palavra representando o nome do candidato, um número inteiro indicando a idade do candidato em anos no momento de realização do concurso e cinco valores em ponto flutuante representando cada uma das notas do candidato em uma das avaliações. As notas vão se 0,0 (zero) a 10,0 (dez). As informações são separadas por espaços em branco.

Saída

A saída é composta por N linhas contendo, cada uma contendo o nome de um candidato. A ordem de apresentação dos nomes segue a classificação no concurso. Ou seja, a primeira linha apresenta o primeiro colocado, a segunda linha o segundo colocado e assim por diante.

Exemplo

Entrada	Saída
1.0 1.0 3.0 2.5 2.5 6 Ana 30 8.3 4.5 9.2 4.0 6.6 João 25 2.0 3.4 8.9 7.2 4.4 Pedro 30 7.8 5.0 9.2 6.0 4.6 Maria 28 5.0 6.0 7.0 5.0 4.0 Thiago 40 6.5 4.5 7.0 4.5 4.5 Raquel 26 10.0 10.0 10.0 10.0 10.0	Raquel Ana Pedro João Thiago Maria

Dica

Reveja o comportamento dos operadores relacionais apresentados na Aula 2 quando aplicados sobre strings.

2ª Questão (3,0 pontos)

Corretores de texto embutidos em aplicativos de mensagens são ferramentas bastante úteis. Uma pesquisa recente observou que um erro bastante comum cometido por usuários que utilizam esses aplicativos em ônibus, metrôs e trens lotados é a digitação repetida do fim das palavras. Por exemplo, ao invés de digitar “abacaxi”, muitas pessoas digitam “abacaxiaxi”. O motivo talvez seja o empurra-empurra dos passageiros.

Escreva um programa que elimina as repetições encontradas nas palavras que compõem as mensagens enviadas pelo usuário.

As mensagens são informadas para seu programa a partir de um arquivo texto de entrada e seu programa deve emitir a resposta em um arquivo texto de saída.

Entrada

O nome do arquivo texto de entrada é “mensagens_originais.txt”. A primeira linha do arquivo de entrada indica a quantidade N de mensagens. Cada uma das N linhas seguintes contém uma mensagem composta por uma ou mais palavras. As palavras são separadas por um espaço em branco e as mensagens não incluem pontuação. Todas as palavras são compostas por letras minúsculas, sem acento e sem cedilha. As repetições ao término das palavras podem ser de qualquer tamanho. As mensagens não incluem palavras que naturalmente são compostas por repetições no final, tais como “banana” e “arara”.

Saída

O nome do arquivo texto de saída é “mensagens_corrigidas.txt”. Seu programa deve escrever N linhas no arquivo de saída, cada uma contendo a quantidade de palavras corrigidas, seguida por um espaço em branco e pela versão corrigida da mensagem de entrada.

Exemplo

Entrada (`mensagens_originais.txt`)	Saída (`mensagens_corrigidas.txt`)

`3`
`estou indodo para a aulaula`
`aa provaova estavatava facil`
`programar requer pratica`
	`2 estou indo para a aula`
	`3 a prova estava facil`
	`0 programar requer pratica`

3ª Questão (3,0 pontos)

Escreva um programa que recebe dois arquivos texto, um contendo um conjunto de palavras e outro contendo um discurso. O resultado emitido pelo programa deve ser um arquivo texto contendo o número de ocorrências de cada palavra do conjunto no discurso informado.

O detalhe em relação ao texto é que ele foi escrito respeitando o tamanho máximo da linha e a aplicação das regras de translineação. Ou seja, para evitar que uma linha fique mais longa do que a página, quebra-se a última palavra da linha em duas partes e indica-se com um hífen ao fim da primeira parte que a mesma continuará na linha seguinte. Na língua portuguesa a quebra ocorre conforme a divisão silábica da palavra em questão.

Nessa questão você será avaliado com relação ao uso de estruturas que representam conjunto e dicionários, além do uso de arquivos texto, manipulação de strings e contagem. O atendimento aos itens (1) a (4) é obrigatório. Código fonte que não segue a especificação não será considerado na correção dos quesitos associados a esses itens:

1) O nome do primeiro arquivo texto de entrada é “palavras.txt”. Seu conteúdo deve ser lido para dentro de uma estrutura de conjunto (`set`).

2) O nome do segundo arquivo texto de entrada é “discurso.txt”. Seu conteúdo deve ser lido para dentro de uma única string composta pelo texto sem quebras de linha e sem a aplicação das regras de translineação. Por exemplo, o texto dado:

```
este é um exemplo simples de apli-
cação de regras de translineação
```

deverá ser convertido e armazenado na string:

```
este é um exemplo simples de aplicação de regras de translineação
```

3) Após ler o conjunto de palavras e o texto conforme especificado nos itens (1) e (2), seu programa deverá criar uma estrutura de dicionário (`dict`) onde cada item é composto por uma das palavras no conjunto e a contagem de sua ocorrência na string. Restrição: A função `.count` de `str` ou quaisquer outras funções prontas semelhante a essa e disponíveis na API do Python ou em bibliotecas externas à API não poderão ser utilizadas na solução.

4) Após a criação do dicionário especificado no item (3), o programa deverá escrever no arquivo “contagem.txt” o resultado das contagens, seguindo a formatação conforme exemplo.

Entrada

A entrada é composta por:

1) Um arquivo texto de nome “palavras.txt” contendo um conjunto de palavras escritas em letras minúsculas. Cada palavra ocupa uma linha desse arquivo.

2) Um arquivo texto de nome “discurso.txt”, contendo um texto qualquer composto por várias linhas e que faz uso de regras de translineação. O texto não possui caracteres de pontuação, as palavras são escritas todas em letras minúsculas e o texto não contém palavras hifenizadas (por exemplo, guarda-chuva).

Saída

Seu programa deve emitir o arquivo “contagem.txt”, onde cada linha é composta por uma palavra do conjunto de entrada, seguida por um espaço em branco e um valor inteiro indicando o número de ocorrências dessa palavra no discurso informado.

Exemplo

Entrada (`palavras.txt`)	Entrada (`discurso.txt`)	Saída (`contagem.txt`)
teste de mesa acabaxi bastante de	para validar minha solução foi preciso fazer o teste de mesa do algoritmo elaborado só depois do teste de mesa que o programa foi implementado essa estratégia poupou bastan- te tempo de desenvolvimento	acabaxi 0 teste 2 mesa 2 bastante 1 de 3

Dica

Utilize subprogramação para organizar seu programa. Por quebrar um problema difícil em problemas mais simples, subprogramação ajuda a organizar tanto o código quando das ideias.

4ª Questão (2,0 pontos)

Dado um arquivo binário contendo valores inteiros e valores em ponto flutuante, escreva um programa que substitua valores existentes no arquivo por novos valores informados pelo usuário. De forma mais específica, o usuário informará via entrada padrão um valor inteiro e um valor em ponto flutuante, e seu programa deverá:

1) Percorrer os registros contidos no arquivo e substituir todos os valores inteiros que sejam menores que o valor informado pelo usuário por -1.

2) Percorrer os registros contidos no arquivo e substituir todos os valores em ponto flutuante que sejam maiores que o valor informado pelo usuário por 9999.0.

O arquivo binário obrigatoriamente atende ao seguinte formato: um valor inteiro de 4 bytes indicando a quantidade de registros armazenados, seguido por registros de 8 bytes cada, sendo os 4 primeiros bytes de cada registro um valor inteiro e os 4 últimos bytes de cada registro um valor em ponto flutuante.

Entrada

A entrada é composta por um arquivo binário de nome "valores.bin", que segue o formato especificado anteriormente, e por dois valores informado via entrada padrão, um valor inteiro (`int`) e um valor em ponto flutuante (`float`).

Saída

Conforme indicado anteriormente neste enunciado, a saída é definida sobre o próprio arquivo de entrada "valores.bin". Ela consiste no arquivo de entrada modificado, onde os valores inteiros e os valores em ponto flutuante são substituídos conforme as regras (1) e (2).

Exemplo

O exemplo que segue mostra o conteúdo do arquivo binário na entrada e na saída de forma textual apenas para facilitar a interpretação dos dados. Você deve estar ciente que o arquivo é binário, tendo cada valor armazenado em 4 bytes e sem espaços em branco, conforme a especificação.

Entrada (arquivo `valores.bin` antes de ser modificado)	Entrada Padrão
`5 -9 20.5 6 10.8 10 8.0 45 -99.6 12 -54.7`	`10` `9.0`

Saída (arquivo `valores.bin` após ser modificado)
`5 -1 9999.0 -1 9999.0 10 8.0 45 -99.6 12 -54.7`

Observações

Se a questão for resolvida considerando arquivos texto então a nota atribuída para a mesma será 0 (zero), mesmo que a solução esteja correta no contexto de arquivos texto.

É proibido ler o conteúdo de todo o arquivo para a memória principal simultaneamente. A substituição de valores deve acontecer dentro do arquivo. Será atribuída nota 0 (zero) para soluções que não atendam a essa restrição, mesmo que a solução esteja correta no contexto de manipulação de listas.

Dicas

Para fins de depuração de código, escreva um programa auxiliar que crie arquivos binários contendo nada mais do que valores armazenados conforme a especificação do enunciado.

Escreva, também, programas auxiliares que ajudem a verificar se o conteúdo do arquivo de saída está correto. Os programas auxiliares não devem ser entregues junto com a solução da questão. Caso sejam entregues, os mesmos não serão considerados na correção, independentemente de estarem corretos ou errados.

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, independente da plataforma.

Boa Avaliação!