Fundação CECIERJ - Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AP3 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 uma pessoa (`str`). Caso o nome da pessoa não seja uma string vazia leia da entrada padrão o sexo (`str`), idade (`int`), altura (`float`) e peso (`float`) dela. Repita esta iteração até que um nome vazio seja digitado. A cada iteração inclua em um dicionário um novo par (chave, valor), cuja chave é o nome da pessoa e o valor uma tupla contendo os demais dados informados. Ao final, após todos os pares serem lidos, mostre o conteúdo do dicionário produzido. Caso o dicionário esteja vazio, escreva a mensagem: "Nenhuma pessoa foi inserida no dicionário!". Veja os exemplos a seguir e reproduza a saída solicitada, a cada caso de teste.

Exemplos

Entrada	Saída
<linha deixada em branco>	`Nenhuma pessoa foi inserida no dicionário!`

Entrada	Saída
`Maria da Silva` `feminino 34 1.70 65.8` `Maria da Silva` `feminino 34 1.33 69.8` `Maria da Silva` `feminino 34 1.66 60.1` *<linha deixada em branco>*	`Dicionário de Pessoas Inseridas:` `Maria da Silva feminino 34 1.66 60.1`

Entrada	Saída
Lia da Silva feminino 34 1.70 65.8 Juca de Sá masculino 44 1.76 95.3 Antônio Lopes Frias masculino 35 1.61 55.5 Mara Menezes Castro feminino 62 1.57 49.0 <linha deixada em branco>	Dicionário de Pessoas Inseridas: Antônio Lopes Frias masculino 35 1.61 55.5 Juca de Sá masculino 44 1.76 95.3 Lia da Silva feminino 34 1.7 65.8 Mara Menezes Castro feminino 62 1.57 49.0

Distribuição de Pontos

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

2ª Questão (3,5 pontos)

Suponha o programa principal a seguir:

```
# Programa Principal
placaVeiculo = input()
nomeArquivoMultas = input()
nomeArquivoValoresMultas = input()
cruza(placaVeiculo, nomeArquivoMultas, nomeArquivoValoresMultas)
```

Implemente o procedimento `cruza`, que processa a placa do veículo lida, dada na string em `placaVeiculo`, observando o conteúdo do arquivo de ocorrências de multas, dado pela string em `nomeArquivoMultas`, considerando o arquivo que contém os valores de cada multa, dado pela string em `nomeArquivoValoresMultas`. Os arquivos texto têm os seguintes formatos:

Arquivo de ocorrências de multas:

```
placa código data horário local
```

Exemplo de uma linha de um arquivo destes:

```
LLP0101 veloz13 01/02/2019 13:45 Angra
```

Arquivo de valores (em reais) das multas, para cada código de multa:

```
codigo valor
```

Exemplo de uma linha de um arquivo destes:

```
veloz13 150.80
```

O procedimento `cruza` deve percorrer todo o arquivo de ocorrências de multa, dado por `nomeArquivoMultas`, e produzir uma lista de todas as multas do veículo, cuja `placaVeiculo` foi informada. A cada multa do veículo deve-se consultar, em `nomeArquivoValoresMultas`, e informar, via escrita na saída padrão, o valor devido. Ao final, escreva o total devido pelo proprietário do veículo. Caso nenhuma multa seja encontrada escreva a mensagem: “Veículo de Placa”, `placaVeiculo`, “não possui multas!!!”.

Exemplos

Arquivo `testeOcorrencias.txt`
LLP0101 V13 01/02/2019 13:45 Angra LLP0101 V33 15/03/2019 16:30 Rio XLX0202 P10 05/01/2019 14:00 Saquarema

Arquivo `tabelaValores.txt`
`V13 130.50` `V33 190.22` `P60 180.00` `P80 400.00`

Entrada	
Entrada Padrão	Saída Padrão
`XRG1000` `testeOcorrencias.txt` `tabelaValores.txt`	`Veículo de Placa XRG1000 não possui multas!!!`

Entrada	
Entrada Padrão	Saída Padrão
`LLP0101` `testeOcorrencias.txt` `tabelaValores.txt`	`Relação de Multas do Veículo com Placa LLP0101` `-----------------------------------------------` `('V13', '01/02/2019', '13:45', 'Angra') R$ 130.50` `('V33', '15/03/2019', '16:30', 'Rio') R$ 190.22` `-----------------------------------------------` `Total das Multas: R$ 320.72`

Distribuição de Pontos

Leitura de arquivos texto e conversão de tipos – 1,0 ponto; Cruzamento de dados – 2,0 ponto; Formatação e impressão da saída – 0,5 pontos.

3ª Questões (4,0 pontos)

Considere a existência de um arquivo binário chamado “bagunca.bin”. Os primeiros 4 bytes desse arquivo armazenam um valor inteiro *N* que indica quantos registros o arquivo contém. Cada registro é formado por três valores numéricos de ponto flutuante, cada valor composto por 4 bytes. O problema é que os registros estão fora de ordem e é sua obrigação ordená-los.

Escreva um programa que abra o arquivo indicado, leia a quantidade de registros contidos nele e ordene os *N* registros conforme o seguinte critério:

> Seja a_1, a_2 e a_3 os três valores que compõe um registro *A* qualquer e b_1, b_2 e b_3 os três valores que compõe um registro *B* qualquer, para $A \neq B$. O registro *A* deverá anteceder *B* na ordenação se a_1 for menor que b_1. Em caso de empate, o registro *A* deverá anteceder *B* na ordenação se a_2 for menor que b_2. Finalmente, em caso de novo empate, o registro *A* deverá anteceder *B* na ordenação se a_3 for menor que b_3.

Utilize o algoritmo *Selection Sort* na ordenação. Este algoritmo é baseado em se passar sempre o registro de menor importância para a primeira posição, depois o de segunda menor importância para a segunda posição, e assim é feito sucessivamente com os registros restantes, até os últimos dois registros.

Você deve utilizar subprogramação na solução apresentada. Não é permitido manter o conteúdo de todo o arquivo armazenado ao mesmo tempo na memória principal, isto é, apenas os dois registros que estão sendo comparados a cada iteração do algoritmo *Selection Sort* podem estar carregados na memória em um dado instante.

Exemplo

Se antes de ser ordenado o arquivo “bagunca.bin” contém os valores

4	3.5	6.7	3.4	3.5	5.2	1.0	1.0	5.6	3.4	9.0	8.5	1.2

então após a ordenação o arquivo conterá os valores

4	1.0	5.6	3.4	3.5	5.2	1.0	3.5	6.7	3.4	9.0	8.5	1.2

Observações

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

É proibida a chamada a implementações prontas de métodos de ordenação disponíveis na API do Python 3. Será atribuída nota 0 (zero) para soluções que façam tais chamadas.

Caso seja implementado um método de ordenação diferente do *Selection Sort*, então a pontuação desse quesito será de 1,0 ponto ao invés de 2,0 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.

Distribuição de Pontos

Leitura e escrita de arquivo binário – 2,0 pontos; Ordenação – 2,0 ponto.

Boa Avaliação!