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

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AP2 2° semestre de 2016

IMPORTANTE

- Prova sem consulta e sem uso de qualquer aparato eletrônico.
- Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas 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)

Utilizando subprogramação, faça um programa Python que processe arquivos texto. Seu programa deve pedir ao usuário o nome de um arquivo texto a ser processado, produza e mostre o conjunto (estrutura de dados "**set**", conforme vídeo-aula 10) de palavras que ocorrem no arquivo. Este arquivo deverá ser percorrido apenas uma vez. Não é permitido manter o conteúdo de todo o arquivo armazenado ao mesmo tempo na memória principal, isto é, apenas uma linha do arquivo pode estar em processamento em um dado instante. Nesta questão está **proibido** o uso da estrutura de dado dicionário ("dict").

A saída do seu programa deve mostrar uma palavra a cada linha, conforme exemplo a seguir.

Exemplo

Entrada	Saída

`cores.txt`	`verde`
	`marrom`
	`preto`
	`cinza`
	`branco`
	`vermelho`
	`amarelo`
	`azul`

Conteúdo do Arquivo

cores.txt

`vermelho amarelo azul preto verde`
`marrom azul branco verde vermelho`
`cinza verde`

2ª Questão (4,0 pontos)

Utilizando subprogramação, faça um programa Python que processe arquivos texto. Seu programa deve pedir ao usuário o nome de um arquivo texto a ser processado, produza um dicionário (estrutura de dados "**dict**", conforme vídeo-aula 11) e mostre, ordenadamente pela chave, seu conteúdo na saída padrão. O dicionário produzido deve ter como chaves ("keys") as palavras contidas no arquivo e como valores ("values") as frequências (quantidade de ocorrências) desta palavra no arquivo. Este arquivo deverá ser percorrido apenas uma vez. Não é permitido manter o conteúdo de todo o arquivo armazenado ao mesmo tempo na memória principal, isto é, apenas uma linha do arquivo pode estar em processamento em um dado instante.

A saída do seu programa deve mostrar uma palavra e sua respectiva frequência a cada linha, conforme exemplo a seguir.

Exemplo

Entrada	Saída

`cores.txt`	`amarelo ocorreu 1 vez(es)`
	`azul ocorreu 2 vez(es)`
	`branco ocorreu 1 vez(es)`
	`cinza ocorreu 1 vez(es)`
	`marrom ocorreu 1 vez(es)`
	`preto ocorreu 1 vez(es)`
	`verde ocorreu 3 vez(es)`
	`vermelho ocorreu 2 vez(es)`

Conteúdo do Arquivo

cores.txt

`vermelho amarelo azul preto verde`
`marrom azul branco verde vermelho`
`cinza verde`

3ª Questão (3,5 pontos)

Considere a existência de um arquivo binário chamado "bagunca.bin". Os primeiros 4 bytes desse arquivo armazenam um valor inteiro primitivo *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

Boa Avaliação!