Fundação CECIERJ - Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AD2 2° semestre de 2016

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. Serão aceitos apenas soluções escritas na linguagem Python 3. Respostas entregues fora do formato especificado, por exemplo, em arquivos com extensão ".pdf", ".doc" ou outras, não serão corrigidas.
- Faça uso de boas práticas de programação, em especial, na escolha de identificadores de variáveis e subprogramas, e comentários no código.
- As ADs deverão ser entregues pela atividade "Entrega de AD2" antes da data final de entrega estabelecida no calendário de entrega de ADs.
- A AD é 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.
-

1ª Questão (2,0 pontos)

Considerando as operações **cons**, **car** e **cdr** vistas na Aula 7, sem utilizar a estruturas de controle de repetição (for e while), sequência (suite com mais de uma linha de comando), nem fatiamento, faça os subprogramas especificados a seguir:

```
def obterSetor(lista, inicio, fim):
      # função que recebe como parâmetros uma lista elementos e
      # um par de valores inteiros, chamados de inicio e de fim,
      # que definem o intervalo de elementos da lista que devem
      # compor o retorno da função.
```

Exemplos

Parâmetros de Entrada: lista, inicio, fim
["ana", "maria", "chico", "igor", "juca"], 2, 4 ["ana", "maria", "chico", "igor", "juca"], 4, 1 [10, 2, 5, 13, 26, 4, 2, 9, 33, 18, 6, 99, 12, 17], 5, 9 [10, 2, 5, 13, 26, 4, 2, 9, 33, 18, 6, 99, 12, 17], 15, 19
Retorno da Função
["maria", "chico", "igor"] [] [13, 26, 4, 2, 9, 33] []

```
def rodar(lista, nVezes):
      # função que rotaciona nVezes os valores da lista no
      # sentido horário, se o número inteiro nVezes for positivo,
      # ou rotaciona nVezes os valores da lista no sentido
      # anti-horário, se nVezes for negativo.
```

Exemplos

Parâmetros de Entrada: lista, nVezes
["ana", "maria", "chico", "igor", "juca"], 2
["ana", "maria", "chico", "igor", "juca"], -2
[10, 2, 5, 13, 26, 4, 2, 9, 33, 18, 6, 99, 12, 17], 4
[10, 2, 5, 13, 26, 4, 2, 9, 33, 18, 6, 99, 12, 17], -15
Retorno da Função
["igor", "juca", "ana", "maria", "chico"]
["chico", "igor", "juca", "ana", "maria"]
[6, 99, 12, 17, 10, 2, 5, 13, 26, 4, 2, 9, 33, 18]
[2, 5, 13, 26, 4, 2, 9, 33, 18, 6, 99, 12, 17, 10]

```
def ordenar(lista):
      # Utilizando o seguinte método de ordenação, chamado de
      # Ordenação por Inserção ("Insertion Sort"), observe o
      # estado da lista recebida como parâmetro, caso esteja
      # vazia, sua função deve retornar uma lista vazia. Caso a
      # lista não esteja vazia, o retorno da função ordenar deve
      # ser a chamada uma a uma função coadjuvante, chamada aqui
      # de insereOrdenado, contendo dois parâmetros: o valor a
      # ser inserido, isto é o car(lista), e uma lista ordenada
      # de todos os demais elementos da lista, isto é o
      # ordenar(cdr(lista)), que insere ordenadamente o primeiro
      # valor da lista recebida como parâmetro numa lista que
      # seja a ordenação de todos os demais valores da lista
      # recebida como parâmetro.
```

Exemplos

Parâmetro de Entrada: lista
[]
["ana", "maria", "chico", "igor", "juca"]
[10, 2, 5, 13, 26, 4, 2, 9, 33, 18, 6, 99, 12, 17]
Retorno da Função
[]
["ana", "chico", "igor", "juca", "maria"]
[2, 2, 4, 5, 6, 9, 10, 12, 13, 17, 18, 26, 33, 99]

2ª Questão (1,5 pontos)

Faça um programa que leia do usuário dois nomes de arquivos, um de referência e um de pesquisa, que poderão ser percorridos apenas uma vez cada. Faça subprogramas que:

a) Leia o conteúdo do arquivo de referência e produza um conjunto das palavras contidas nele.

b) Receba como parâmetro o conjunto de palavras de referência e o nome do arquivo de pesquisa, e escreva, na saída padrão, a palavra e seu posicionamento no arquivo de pesquisa caso ela esteja no conjunto de referência.

Não é permitido manter o conteúdo de todo o arquivo armazenado ao mesmo tempo na memória principal.

Entrada

A entrada é composta de duas linhas, cada uma contendo o nome de um arquivo texto a ser utilizado.

Saída

Para palavra do arquivo de pesquisa que aconteça no conjunto de palavras de referência, uma linha deve ser escrita na saída padrão, no seguinte formato:

(palavra, número da linha no arquivo de pesquisa, número da palavra nesta linha)

Exemplo

Entrada	Saída

`cores.txt`
`carta.txt`
	`("amarelo", 3, 3)`
	`("azul", 3, 5)`
	`("verde", 4, 3)`

Conteúdo dos Arquivos

cores.txt	carta.txt

`vermelho amarelo azul preto verde`
`marrom azul branco vermelho`
`cinza`
	`O tempo era curto`
	`Mas ele insistiu`
	`Misturando o amarelo com azul`
	`Criou o verde esperanca`

3ª Questão (1,5 pontos)

Considere a existência de dois arquivos texto: o primeiro que representa uma coleção de palavras onde cada palavra é associada a um sinônimo, e o segundo contendo um texto livre a ser processado.

Escreva um programa que:

a) No programa principal, solicite ao usuário que informe o nome do arquivo de sinônimos e o nome do arquivo contendo o texto a ser processado. As mensagens a serem emitidas são, respectivamente:

"Informe o nome do arquivo de sinônimos: "

"Informe o nome do arquivo com o texto: "

b) Em seguida, o programa principal deve ativar a função cujo comportamento deverá ser implementado conforme a descrição abaixo:

```
def lerSinonimos(nomeArq):
      # Retorna um dicionário composto por pares chave/valor,
      # lidos do arquivo de nome informado como parâmetro. Cada
      # linha do arquivo contém um par de palavras. A primeira
      # palavra corresponde à chave do par e a segunda é o
      # sinônimo, tratado aqui como valor associado à chave.
      # Lembre-se de sair do subprograma apenas depois de fechar
      # o arquivo.
```

c) Por fim, após obter o dicionário de sinônimos, o programa principal deverá ativar a rotina que processa o segundo arquivo conforme a especificação abaixo:

```
def processarTexto(nomeArq, sinonimos):
      # Abre o arquivo texto de nome informado e copia seu
      # conteúdo para um arquivo texto temporário. Entretanto,
      # toda palavra do arquivo informado que conste como chave
      # no dicionário de sinônimos deverá ser substituída por seu
      # respectivo sinônimo no momento da escrita.
      # É importante observar que espaços em branco e quebras
      # de linha devem ser preservados no arquivo a ser gerado.
      # Além disso, o arquivo deverá ser percorrido apenas uma
      # vez e não é permitido manter o conteúdo de todo o arquivo
      # armazenado ao mesmo tempo na memória principal.
      # Após o arquivo temporário ser criado, o arquivo original
      # deverá ser sobrescrito pelo temporário por cópia
      # explícita de cada linha. Novamente, não é permitido
      # manter o conteúdo de todo o arquivo armazenado ao mesmo
      # tempo na memória principal e deve-se ter cuidado para
      # preservar espaços em branco e quebras de linha.
      # Lembre-se de sair do subprograma apenas depois de fechar
      # ambos os arquivos.
```

É garantido que ambos os arquivos, o de sinônimos e o a ser processado, contém apenas palavras escritas com letras minúsculas e sem o uso de pontuação.

Na rotina descrita no item (c), você não precisa se preocupar a conjugação verbos nem com a adequação de singular e plural na substituição das palavras. Ou seja, a troca de palavras no texto pelo seu sinônimo ocorre apenas se a palavra a ser trocada aparece exatamente como descrita no dicionário de sinônimos.

Exemplo

Entrada Digitada pelo Usuário
sinonimos.txt texto.txt

Conteúdo dos Arquivos

sinonimos.txt
seguir continuar problemas adversidades adocar acucarar crer acreditar superar vencer

texto.txt (antes do processamento)
e importante crer que mesmo diante de grandes problemas somos capazes de superar nossas limitacoes e seguir no caminho que leva ao sucesso

texto.txt (depois do processamento)
e importante acreditar que mesmo diante de grandes adversidades somos capazes de vencer nossas limitacoes e avancar no caminho que leva ao sucesso

4ª Questão (1,5 pontos)

Faça um programa que:

a) Considere a existência de um arquivo texto que defina turmas em uma universidade. Na primeira linha é dada a quantidade *N* de turmas. Nas *N* linhas seguintes é dado o endereço do arquivo que descreve cada turma.

b) Cada arquivo texto que descreve os resultados de uma turma possui na sua primeira linha o número *A* de alunos da turma. Nas *A* linhas seguintes é dada matrícula do aluno e sua respectiva nota.

Nessa questão, implemente as operações:

```
def lerTurmas(nomeArq):
      # Que retorna uma lista de turmas, cada uma sendo
      # uma lista de tuplas (matricula, nota)

def calcularMedia(listaDeListas, mat):
      # Que retorna a média de todas as notas obtidas
      # pelo aluno de matrícula mat.
```

Entrada

A primeira linha define o nome do arquivo de turmas a ser processado.

A segunda linha define a matrícula do aluno que se deseja calcular a média das notas.

Saída

Imprima a matrícula do aluno, seguida do texto "obteve média" e do valor da média computada.

Exemplo

Entrada	Saída
TecnologiaEmSistemasDeComputacao.txt 216031013	216031013 obteve média 6.8

Conteúdo dos Arquivos

TecnologiaEmSistemasDeComputacao.txt	EstruturasDeDados.txt
3 FundamentosDeProgramacao.txt EstruturasDeDados.txt FAC.txt	2 116031082 3.4 216031013 6.5

FundamentosDeProgramacao.txt	FAC.txt
4 115031246 2.8 214031331 7.8 216031013 8.5 115031444 8.1	3 116031082 3.4 216031013 5.4 115031255 4.9

5ª Questão (1,5 pontos)

Escreva um programa que:

a) Solicite ao usuário o nome de um arquivo binário que mantém uma sequência de valores inteiros não negativos, dispostos em ordem crescente. Cada valor é armazenado em um inteiro primitivo de 4 bytes. A mensagem a ser emitida é:

"Informe o nome do arquivo binário de números: "

b) Em seguida, abra o arquivo, mas não leia seu conteúdo de uma só vez para a memória principal. Inclusive, nessa questão é proibido fazer a carga completa do arquivo, pois deve-se assumir que o arquivo é tão grande que a leitura completa levaria à falta de memória e ao término prematuro de seu programa.

c) Após abertura do arquivo, seu programa deverá entrar em uma repetição que, a cada iteração, solicita ao usuário que informe um número a ser localizado no arquivo. O programa deverá emitir a seguinte mensagem na solicitação:

"Favor informar o valor a ser encontrado (-1 para sair): "

Caso o valor informado seja -1, a repetição termina e seu programa é encerrado com sucesso. Caso contrário, seu programa utilizará a função "buscar", para retornar a posição do número solicitado, caso esse exista no arquivo, ou o valor -1 caso esse não exista. O valor retornado pela função deverá ser impresso para o usuário na saída padrão com a mensagem:

"O número está na posição %d\n"

ou

"O número não foi encontrado.\n"

Abaixo é apresentada a especificação da função "buscar":

```
def buscar(arq, num):
      # Utiliza deslocamento de cursos (Aula 12) e busca binária
      # para localizar no arquivo binário arq o número num
      # informado. A função retorna a posição do número no
      # arquivo, sendo que esta vai de zero à quantidade de
      # números armazenados menos 1, ou -1 caso o número não
      # conste no arquivo.
```

d) No programa principal, feche o arquivo após sair da repetição.

Dica: Para fins de depuração de código, escreva um programa auxiliar que crie arquivos binários contendo nada mais do que valores inteiros armazenados sequencialmente conforme a especificação do enunciado, como inteiros primitivos de 4 bytes. O programa auxiliar não deve ser entregue junto com a solução da questão.

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

6ª Questão (2,0 pontos)

Considere a existência de um arquivo binário contendo registros compostos por um valor inteiro primitivo de 4 bytes e dois valores em ponto flutuantes com 4 bytes cada. Escreva um programa que:

a) Solicite ao usuário o nome de um arquivo binário que mantém uma sequência de N registros, montados conforme especificado acima, e também solicite um par de valores, L e H, para $0 \leq L \leq H < N$, que identificam, respectivamente, a posição

inicial e a posição final dos registros do setor a ser processado. As mensagens a serem emitidas são, respectivamente:

"Informe o nome do arquivo binário: "

"Informe a posição inicial do setor a ser processado: "

"Informe a posição final do setor a ser processado: "

b) Em seguida, abra o arquivo, mas não leia seu conteúdo de uma só vez para a memória principal. Inclusive, nessa questão, é proibido fazer a carga completa do arquivo ou do setor informado, pois deve-se assumir que o arquivo é tão grande que a leitura completa do setor ou do arquivo levaria à falta de memória e ao término prematuro de seu programa.

c) Após abertura do arquivo, seu programa deverá invocar um subprograma que imprime na saída padrão o conteúdo de cada registro. Esse subprograma deverá atender à seguinte especificação

```
def imprimir(arq):
      # Percorre do primeiro ao último registro armazenado em
      # arq, imprimindo na saída padrão um registro por linha e
      # no formato (v1, v2, v3), onde v1 é o valor inteiro e v2 e
      # v3 são os valores em ponto flutuante a serem exibidos com
      # duas casas decimais.
```

d) De volta ao programa principal, seu programa deverá invocar um subprograma que ordene os registros do setor informado. Esse subprograma deverá atender à seguinte especificação:

```
def ordenar(arq, l, h):
      # Utiliza deslocamento de cursos (Aula 12) e Ordenação por
      # Inserção ("Insertion Sort"), o setor de registros de
      # posição l a h deverão ser ordenados de forma crescente de
      # valores v1. Caso ocorra empate nos valores v1 de um par
      # de registros, o valor v2 deve ser considerado e, em caso
      # de um novo empate, o valor v3.
```

e) Após a ordenação, o programa principal deverá invocar novamente o subprograma "imprimir", especificado no item (c).

f) Feche o arquivo após exibir o conteúdo já ordenado.

Dica: Para fins de depuração de código, escreva um programa auxiliar que crie arquivos binários contendo nada mais do que os registros armazenados sequencialmente conforme a especificação do enunciado. O programa auxiliar não deve ser entregue junto com a solução da questão.

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

Boa Avaliação!