Fundação CECIERJ - Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AP3 1° semestre de 2018

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 (3,5 pontos)

Implemente a função `ordenaPorCampo`, cujo cabeçalho é:

```
def ordenaPorCampo(qualCampo, vetorDeProdutos):
```

O argumento/parâmetro `qualCampo` é um número inteiro no intervalo 1 até 4, que define a informação que será utilizada para ordenar o vetor de produtos. O argumento/parâmetro `vetorDeProdutos`, é um vetor de zero ou mais produtos, onde cada produto é expresso pela lista com quatro informações, conforme descrito a seguir:

```
[código, descrição, quantidade, preço]
```

Exemplos de produtos:

```
["xy3", "arroz", 120, 5.25]
["aa9", "feijão", 555, 2.99]
["ab8","banana", 100,3.99]
```

Exemplo de vetor de produtos:

```
lista = [["xy3", "arroz", 120, 5.25],
         ["aa9", "feijão", 555, 2.99],
         ["ab8","banana", 100, 3.99]]
```

A função `ordenaPorCampo` deve produzir e retornar um novo vetor, que ordena o conteúdo do `vetorDeProdutos` pelo campo escolhido. Se `qualCampo = 1`, ordenar pelo código, se `qualCampo = 2`, ordenar pela descrição, se `qualCampo = 3`, ordenar pela quantidade e, finalmente, se `qualCampo = 4`, ordenar pelo preço.

Exemplos de Resultado

Lista retornada no caso de uma chamada à `ordenaPorCampo(1, lista)`:

```
[["aa9", "feijão", 555, 2.99],
 ["ab8", "banana", 100, 3.99],
 ["xy3", "arroz", 120, 5.25]]
```

Lista retornada no caso de uma chamada à `ordenaPorCampo(2, lista)`:

```
[["xy3", "arroz", 120, 5.25],
 ["ab8", "banana", 100, 3.99],
 ["aa9", "feijão", 555, 2.99]]
```

Lista retornada no caso de uma chamada à `ordenaPorCampo(3, lista)`:

```
[["ab8", "banana", 100, 3.99],
 ["xy3", "arroz", 120, 5.25],
 ["aa9", "feijão", 555, 2.99]]
```

Restrições

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, sua função deverá implementar um dos métodos de ordenação vistos na Aula 7, sendo atribuída nota 0,0 (zero) soluções que violem tal restrição.

Você pode implementar rotinas auxiliares. Entretanto, a definição da assinatura da função `ordenaPorCampo` deve ser idêntica à apresentada no enunciado desta questão. Ou seja, não inclua ou remova argumentos /parâmetros dessa rotina.

Distribuição de Pontos

Implementação da ordenação – 2,5 pontos; Não repetir o código de ordenação para cada campo – 1,0 ponto.

2ª Questão (3,0 pontos)

Implemente o subprograma `insereOrdenado`, cujo cabeçalho é:

```
def insereOrdenado(nomeDoArquivo, novoValor):
    ...
    return None
```

Suponha que o arquivo texto recebido no primeiro parâmetro, isto é, em `nomeDoArquivo`, tem um número de ponto flutuante em cada linha. Além disso, suponha que o conteúdo do arquivo já está ordenado crescentemente. O que o subprograma `insereOrdenado` deve fazer é inserir o `novoValor`, recebido no segundo parâmetro/parâmetro, no arquivo texto, de forma que o arquivo preserve sua ordenação.

Exemplo

Para o arquivo informado, de nome “teste.txt” e conteúdo:

Arquivo de exemplo “teste.txt” antes da inserção
4.6 5.12 5.12 6.8 8.5 8.67

O efeito da chamada `insereOrdenado("teste.txt", 6.0)` atualiza o arquivo para:

Arquivo de exemplo “teste.txt” após a inserção
4.6 5.12 5.12 6.0 6.8 8.5 8.67

Restrição

Não é permitido manter todo o conteúdo do arquivo em memória principal, sendo atribuída nota 0,0 (zero) para soluções que violem tal restrição.

Sugestão

Utilize um arquivo auxiliar.

Distribuição de Pontos

Leitura dos dados existentes no arquivo – 1,0 pontos; Inserção na posição adequada – 1,0 ponto; Escrita do arquivo atualizado – 1,0 pontos.

3ª Questão (3,5 pontos)

Leia de um arquivo binário uma coleção de N pares de valores inteiros e calcule o máximo divisor comum (MDC) desses valores pela ativação da função recursiva `mdc`, de cabeçalho:

```
def mdc(a, b):
```

e imprima na saída padrão os MDCs calculados. Por definição, o MDC de dois ou mais números inteiros é o maior número inteiro que é fator de tais números.

Mais especificamente, seu programa deverá:

a) Abrir e ler o conteúdo do arquivo binário “valores.bin”, composto por um valor inteiro N que indica a quantidade de pares de valores inteiros não negativos (A e B) e os valores A e B propriamente ditos.

b) Implementar a função recursiva `mdc`, que é definida conforme o Algoritmo Euclidiano por:

`mdc(a, b) == mdc(b, a % b)`, para `b > 0` e

`mdc(a, 0) == a`, caso contrário.

c) Ativar a função `mdc` para cada um dos N pares de valores A e B lidos do arquivo e imprimir o resultado na saída padrão, colocando um resultado por linha.

Exemplo

Assumindo, neste exemplo, que o arquivo “valores.bin” é composto pelos valores:

Conteúdo do arquivo “valores.bin”				
2	12	18	25	26

temos que `N = 2`. Para o primeiro par temos `A = 12` e `B = 18`, enquanto que para o segundo par temos `A = 25` e `B = 26`. Os resultados a serem impressos na saída padrão são:

Saída Padrão
6 1

Dicas

1) Para o primeiro par do exemplo acima, a sequência de chamas recursivas se desenrola da seguinte forma: `mdc(12,18) == mdc(18,12) == mdc(12,6) == mdc(6,0) == 6`.

2) Lembre-se que, por padrão, cada valor inteiro ocupa 4 bytes.

Restrição

Será atribuída nota 0,0 (zero) para o item (a) de soluções que tratarem o arquivo binário como arquivo texto.

Distribuição de Pontos

(a) 1,0 ponto; (b) 2,0 pontos; (c) 0,5 pontos.

Boa Avaliação!