MC102 - Algoritmos e Programação de Computadores

Lista de Exercícios 7

17 de Novembro de 2016

1. Suponha que definimos uma classe para representar produtos de um supermercado:

```
class Produto:
   def __init__(self, p, q):
      self.preco = p
      self.quantidade = q
```

   Implemente duas funções, uma que devolve o vetor ordenado por preços e outra que devolve o vetor ordenado pela quantidade de itens no estoque.

2. Suponha que definimos uma classe para representar Datas:

```
class Data:
   def __init__(self, dia, mes, ano):
      self.dia = dia
      self.mes = mes
      self.ano = ano
```

   Implemente um algoritmo que receba um vetor de Datas como parâmetro e que retorne as datas em ordem cronológica. **Dica:** Ordene o vetor separadamente por cada um dos campos.

3. Suponha que definimos uma classe para representar dados de pessoas e uma outra classe para representar dados de várias pessoas como uma base de dados.

```
class Pessoa:
   def __init__(self, rg, cpf, nome):
      self.rg = rg
      self.cpf = cpf
      self.nome = nome

class Base:
   def __init__(self, armazenado, pessoas):
      self.armazenado = armazenado // Deve sempre corresponder ao
                                   // número de pessoas na base
      self.pessoas = pessoas // Lista de objetos da classe Pessoa
```

   Crie funções para cada uma das operações abaixo:

   - Cria base: esta função devolve uma Base onde o campo **armazenado** é inicializado com 0.

- Inclui Pessoa: esta função recebe como parâmetro um dado do tipo Pessoa e o inclui na base (também passada por parâmetro) caso já não exista na base uma pessoa com o mesmo RG. A função devolve 1 caso a inclusão tenha ocorrido, devolve 0 caso a Base esteja cheia e devolve -1 caso já exista uma pessoa com o RG informado.
- Exclui Pessoa: esta função recebe como parâmetro um dado do tipo int representando o RG de uma pessoa e o exclui da base caso esteja presente. A função devolve 1 caso a exclusão tenha ocorrido, e devolve 0 caso não exista uma pessoa com o RG informado.

4. Suponha que definimos uma classe para representar dados de pessoas e um vetor para armazenar dados de várias pessoas:

```
class Pessoa:
   def __init__(self, rg, cpf, nome):
      self.rg = rg
      self.cpf = cpf
      self.nome = nome

cadastro = [] // lista que armazena pessoas
```

   Suponha que a lista esteja ordenado em ordem crescente por valor de RG. Implemente uma função de busca por RG, que opera como a busca binária, e que caso exista uma pessoa no cadastro com o RG a ser buscado, devolve o índice deste no cadastro, e devolve -1 caso não exista uma pessoa com o RG a ser buscado.

5. Refaça as funções de busca sequencial e busca binária vistas em aula assumindo que o vetor possui chaves que podem aparecer repetidas. Neste caso, você deve retornar em um outro vetor todas as posições onde a chave foi encontrada.

   - Você deve devolver em **posicoes** as posições de **vet** que possuem a **chave**, e devolver em **n** o número de ocorrências da chave.