Lista 6

1. Suponha que criamos uma classe para armazenar produtos de um supermercado:

```
class Produto:
   def __init__(self):
      self.nome=''
      self.preco=0
      self.quantidade=0
```

   Suponha uma lista de objetos da classe `Produto` Implemente duas funções, uma que devolve uma lista ordenada por preços e outra que devolve o vetor ordenada pela quantidade de itens no estoque. Funções **ordenaPreco(lista_prod** e **ordenaQuant(lista_prod)**.

2. Suponha que criamos uma classe para armazenar Datas:

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

   Implemente um algoritmo que receba uma lsita 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 criamos uma classe para armazenar dados de pessoas e um vetor para armazenar dados de várias pessoas:

```
class Pessoa:
   def __init__(self):
      self.rg=0
      self.cpf=0
      self.nome-''
```

Suponha que uma lista de pessoas esteja ordenada 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 cadastro e caso não exista o RG a ser buscado, devolve -1.

4. Refaça as funções de busca sequencial e busca binária vistas em aula assumindo que a lista possui dados que podem aparecer repetidos. Neste caso, você deve retornar uma lista com todas as posições onde a chave foi encontrada.
   Protótipo da função: **busca(lista, chave)**

5. Escreva uma função chamada **teste** que recebe um valor n Sua função deve retornar uma tupla (b, k) de inteiros tal que $b^k = n$ e b seja o menor possível.

6. Faça uma função chamada **primo** que recebe como parâmetro um inteiro m. A função deve retornar a tupla $(p1, p2)$ onde $p1$ é o maior número primo que é menor do que m e $p2$ o menor número primo que é maior do que m.

7. Faça uma função chamada **media** que recebe uma lista e retorna a posição do elemento que tem o valor mais próximo da média.