Oitava Lista de Exercícios - Computational Thinking

1. Escreva um programa que cria uma lista de strings e preenche essa lista com 10 valores que serão digitados pelo usuário. Imprima a lista na tela.

```
dados = []

i = 0
while i < 10:
   valor = input("Digite uma String:")
   dados.append(valor)
   i = i + 1

for info in dados:
   print(info)
```

2. Escreva uma função que recebe como parâmetro um inteiro positivo n e retorna uma lista preenchida com n números inteiros aleatórios entre 1 e 1000.

```
import random

def geraLista(n):
    lista = []
    for i in range(n):
        numero = random.randint(1,1001)
        lista.append(numero)
    return lista
```

3. Escreva um algoritmo que pede para o usuário digitar 10 strings uma de cada vez. Depois que o usuário digitar todas elas, seu programa deverá imprimir as strings na ordem inversa de leitura. Por exemplo se as duas últimas strings foram, respectivamente, avestruz e onça; o programa imprime onça e depois avestruz.

```
dados = []

i = 0
while i < 10:
   valor = input("Digite uma String:")
   dados.append(valor)
   i = i + 1

i = len(dados) - 1
while i >= 0:
   print(dados[i])
   i = i - 1
```

4. Escreva um programa que pede para o usuário digitar um inteiro n. Depois seu algoritmo pede para o usuário digitar uma sequência de n números reais. Após a entrada dos dados, seu programa deverá imprimir os resultados das seguintes somas: $v[0]+v[n-1]$, $v[1]+v[n-2]$, $v[2]+v[n-3]$, ...; até que todos os valores informados tenham participado de alguma soma.

```
dados = []

n = input("Informe a qtd de elementos:")
i = 0
while i < n:
   valor = input("Digite um valor da sequencia:")
   dados.append(valor)
   i = i + 1

i = 0
j = len(dados) - 1
while i <= j:
   print(dados[i] + dados[j])
   i = i + 1
   j = j - 1
```

5. Faça uma função em Python que recebe uma lista de números reais e retorna `True` se a lista está ordenada em ordem crescente ou `False` se ela não está.

```
def isSorted(lista):
   for i in range(1, len(lista)):
       if lista[i-1] > lista[i]:
           return False
   return True
```

6. Escreva uma função em Python que recebe um inteiro x e uma lista de números inteiros ordenada em ordem crescente. Sua função deverá inserir x na lista de forma que ela continue ordenada em ordem crescente. Neste exercício você deve usar apenas o método `insert` da lista.

```
def insereOrdenado(x, lista):
    i = 0
    while i < len(lista) and x < lista[i]:
        i = i + 1

    lista.insert(i, x)

vet = [1, 6, 10, 24, 25, 30, 45]
insereOrdenado(20, vet)
print(vet)
```

7. Escreva uma função que recebe como parâmetro uma lista de números reais e um real x. Sua função deverá contar e retornar a quantidade de elementos que são maiores ou iguais a x.

```
def contagem(x, lista):
    i = 0
    conta = 0
    while i < len(lista):
        if lista[i] >= x:
           conta = conta + 1
        i = i + 1

    return conta
```

8. Escreva uma função que recebe como parâmetro uma lista a de números inteiros e retorna uma outra lista contendo somente os números pares de a.

```
def extraiPares(listaA):
    retorno = []
    i = 0

    while i < len(listaA):
        if listaA[i] % 2 == 0:
           retorno.append(listaA[i])
        i = i + 1

    return retorno
```

9. Escreva uma função que recebe como parâmetro duas listas listaA e listaB de números reais. Ela deverá retornar uma terceira lista contendo todos os números da listaA que também estão na listaB.

```
def interseccao(listaA, listaB):
    resp = []
    for elem in listaA:
        if elem in listaB:
            resp.append(elem)

```

```
    return resp

lA = [2, 5, 7, 10, 8, -1]
lB = [0, 3, 5, 9, 7]
print(interseccao(lA, lB))
```

10. Escreva um método chamado `intercala` que recebe como parâmetro duas listas a e b de números reais ordenadas em ordem crescente. Seu método deverá retornar uma terceira lista contendo a união dos elementos de a e b ordenados em ordem crescente.

```
def intercala(lA, lB):
    resp = []
    i = 0   #percorre lA
    j = 0   #percorre lB
    while i < len(lA) and j < len(lB):
        if lA[i] < lB[j]:
            resp.append(lA[i])
            i = i + 1
        else:
            resp.append(lB[j])
            j = j + 1

    while i < len(lA):
        resp.append(lA[i])
        i = i + 1

    while j < len(lB):
        resp.append(lB[j])
        j = j + 1

    return resp

x = [2, 6, 8, 9, 13, 15, 23]
y = [0, 1, 10, 12]
resultado = intercala(x, y)
print(resultado)
```

11. Na Copa do Mundo do Brasil os quadrifinalistas foram, em ordem alfabética: Alemanha, Argentina, Bélgica, Brasil, Colômbia, Costa Rica, França e Holanda. Imaginando que não sabemos os resultados e nem os cruzamentos, escreva um algoritmo que gere todos os possíveis campeões e vice-campeões dentre os oito times.

```
def imprimeSegundosColocados(primeiro, lista):
   for segundo in lista:
       if segundo != primeiro:
           print(primeiro, segundo)

def imprimeTudo(lista):
   for time in lista:
       imprimeSegundosColocados(time, lista)

times = ["Alemanha", "Argentina", "Belgica", "Brasil", "Colombia", "Costa Rica", "Franca", "
    Holanda"]
imprimeTudo(times)
```

12. Dado uma lista de strings, escreva um algoritmo que conta o número de ocorrências da string na lista. Por exemplo:

```
letras = ['a', 'e', 'b', 'a', 'c', 'a', 'b', 'a', 'e']
```

a: 4 vezes

e: 2 vezes

b: 2 vezes

c: 1 vezes

Dica, crie uma função para ajudar a resolver o problema.

```
#pego um elemento da lista
#se nao foi contabilizado entao
#   faco a contagem das ocorrencias,
#   marco que ele foi contabilizado e
#   imprimo na tela

def contagem(lista):
    #guardarei todos os valores que eu ja contei
    contabilizado = []
    for i in range(len(lista)):
        if not lista[i] in contabilizado:
            qtd = lista.count(lista[i])
            print(lista[i],"aparece:",qtd,"vezes")
            contabilizado.append(lista[i])

contagem(['a', 'b', 'e', 'a', 'a', 'e', 'a', 'b'])
```

13. Considere um corredor com mil portas, numeradas de 1 a 1000, que se encontram todas fechadas. Por esse corredor passarão mil pessoas, que modificarão o estado das portas cujo número seja múltiplo do seu número de passagem: a pessoa com o número 3 modificará o estado (fechará se estiverem abertas ou abrirá se estiverem fechadas) das portas nº 3, 6, 9, 12, ... e a pessoa com o número 7 fará o mesmo às portas 7, 14, 21, etc. Construa um programa que permita saber quantas são e quais são as portas abertas após a passagem da milésima pessoa. Dica: use uma lista onde você armazena valores `True` ou `False` com 1001 posições para representar as portas. Dica, crie uma função para ajudar a resolver o problema.

```
def passagem(p, doors):
    for i in range(1, 1001):
        if i % p == 0:
            if doors[i] == True:
                doors[i] = False
            else:
                doors[i] = True


portas = [False] * 1001

pes = 1
while pes <= 1000:
    passagem(pes, portas)
    pes = pes + 1

for i in range(1, 1001):
    if portas[i] == True:
        print("Porta aberta: ", i)
```

Boa sorte!