**CEFET-RJ - Petrópolis**
**Engenharia de Computação**
Prof. Luis Retondaro

**INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO**

# Lista de Exercícios 2

Esses exercícios são comuns às disciplinas básicas da maioria dos cursos de computação. Portanto, é possível que já existam soluções prontas disponíveis. Evite copiá-las. Assim, você pode melhorar seu desempenho exercitando a escrita de seus próprios algoritmos.

**Questão 1.**
Faça um programa que leia o tempo em segundos e escreva a mesma quantidade de tempo em horas, minutos e segundos.

**Questão 2.**
Defina uma função **triangRetang** que, dados três número inteiros, retorna verdadeiro se e somente se os números podem representar lados de um triângulo retângulo.

Exemplo: **triangRetang**(4,5,3) deve retornar verdadeiro.

Escreva um programa que leia três valores inteiros quaisquer (um em seguida do outro) e imprima uma mensagem, usando a função **triangRetang**, que indica se os valores lidos podem ou não ser lados de um triângulo retângulo.

Você pode supor que os valores lidos são todos inteiros (o seu programa deve funcionar corretamente apenas nesse caso).

**Questão 3.**
Defina uma função que, dados cinco números inteiros, retorna verdadeiro se e somente se o conjunto formado pelos 2 últimos números é um subconjunto daquele formado pelos 3 primeiros.

Escreva um programa que leia 5 valores inteiros, e imprima o resultado de determinar se o conjunto formado pelos 2 últimos é um subconjunto daquele formado pelos três primeiros, usando a função definida acima.

Você pode supor que os valores lidos são todos inteiros (o seu programa deve funcionar corretamente apenas nesse caso).

**Questão 4.**
Escreva um programa que leia uma sequência de valores inteiros e imprima, para cada valor lido, uma mensagem que indica se ele é ou não é um número perfeito.

Um inteiro positivo $n$ é perfeito se é igual à soma dos divisores positivos de $n$ diferentes de $n$. Por exemplo: 6 é perfeito, pois 1+2+3 é igual a 6 (e 1, 2 e 3 são os divisores positivos de 6 diferentes de 6).

A leitura deve terminar quando um valor menor ou igual a zero for lido. (Você deve supor que a entrada consiste apenas de valores inteiros.)

**Questão 5.**
Escreva um programa que leia, do dispositivo de entrada padrão, um $n$ o inteiro positivo $n$, uma cadeia de caracteres de tamanho máximo $n$, e imprima a cadeia em ordem inversa. Não pode utilizar o método ***reverse*** da classe ***list***.

Por exemplo, para a entrada: **10 abcde**
a saída deve ser: **edcba**

**Questão 6.**

Escreva um programa que leia, do dispositivo de entrada padrão, um valor inteiro positivo $n$, em seguida $n$ valores inteiros positivos $v_1, ..., v_n$ , depois um numero inteiro $m$, e imprima, no dispositivo de saida padrão, a soma dos valores $v_1, ..., v_n$ que são maiores que $m$.

Por exemplo, para a entrada: 4 1 2 3 4 2

a saida deve ser: 7 (pois 7 é a soma dos valores 3 e 4, que são os valores maiores que 2 dentre os 4 valores 1, 2, 3, 4).

**Questão 7.**

Faça uma função que leia uma matriz 3 x 3 que representa um tabuleiro de jogo da velha e indique qual posição deveria ser jogada para ganhar o jogo (se possível) ou ao menos para evitar uma derrota.

Faça um programa que utilize esta função para, pelo menos, 3 jogadas seguidas.

**Questão 8.**

**Você conhece o método de ordenação *Selection Sort*?**

- Dê uma olhada nesta explicação: `https://pt.wikipedia.org/wiki/Selection_sort`

**Esse algoritmo faz uso de uma função para selecionar o menor elemento a partir de cada posição do vetor e inserí-lo nesta mesma posição.**

Agora, faça um programa que ordene um vetor de 100 números gerados aleatoriamente, usando o algoritmo Selection Sort. Não utilize o método **sort** da classe *list*.

**Questão 9.**

Faça um programa que leia um número de até 9 dígitos e escreva-o por extenso.

**Questão 10.**

Faça um programa que utilize dois vetores de 50 posições com valores inteiros aleatórios, ordene cada vetor individualmente usando o Selection Sort e combine os dois vetores gerando um novo vetor de 100 posições, de forma que esse novo vetor já seja criado ordenado. Utilize subprogramação (criação de funções) sempre que possível.

**Questão 11.**

Faça um programa que leia uma frase toda em minúsculas e imprima na tela a a mesma frase onde a inicial de cada palavra seja maiúscula.

Exemplo: eu gosto de programar $\longrightarrow$ Eu Gosto De Programar

**Questão 12.**

Jogo da forca.

De posse do arquivo **palavras.py**, faça um programa que escolha aleatoriamente uma das palavras do vetor e peça o usuário para tentar adivinhar a palavra secreta, informando uma letra de cada vez. Assuma que o usuário tem 7 vidas. Ao final mostre a mensagem de vitória caso tenha acertado a palavra secreta, ou se ele foi enforcado.