UNIFEI	Universidade Federal de Itajubá Instituto de Engenharia de Sistemas e Tecnologias da Informação-IESTI
Disciplina	ECOP01 – Técnicas de Programação
Professores	Profª.Drª.Thatyana de Faria Piola Seraphim, Prof.Dr.Enzo Seraphim
7ª Lista de Exercícios	

1) Faça uma função chamada **perfeito** que recebe como parâmetro um número inteiro e retorna verdadeiro ou falso se o número é perfeito ou não. Um número é dito perfeito quando ele é igual a soma dos seus divisores executando ele próprio. (Ex: 6 é perfeito, 6 = 1 + 2 + 3, que são seus divisores). Faça a leitura do número e a chamada da função no programa principal.

2) Faça uma função chamada **contaDigitos** que recebe como parâmetro um número inteiro e retorna a quantidade de dígitos desse número. Faça a leitura do número inteiro e a chamada da função no programa principal.

3) Faça uma função chamada **potencia** que recebe como parâmetro dois valores inteiros e positivos. A função deve calcular e retornar a potência ($base^{expoente}$). Os dois valores inteiros e positivos (*base* e *expoente*) devem ser informados pelo usuário no programa principal. Após a chamada da função, imprima no programa principal o resultado da potência.

4) Faça uma função chamada **harmonico** que recebe como parâmetro um número natural. A função deve retornar o número harmônico H_n definido por: $H_n = 1 + 1/2 + 1/3 + 1/4 + ... 1/n$. O número deve ser informado pelo usuário no programa principal. Após a execução da função, imprima o valor retornado pela função no programa principal.

5) Faça uma função chamada **primo** que recebe como parâmetro um número natural *n* e retorna se o número é primo ou não. Para identificar se o número é primo ou não, verifique se ele tem raiz quadrada inteira. O número deve ser informado pelo usuário no programa principal. Após a execução da função, imprima o valor retornado pela função no programa principal.

6) Faça uma função chamada **hipotenusa** que recebe como parâmetro dois valores inteiros que representam os dois lados de um triângulo retângulo. A função deve retornar o valor da hipotenusa desse triângulo. Os valores inteiros que representam os lados devem ser informados pelo usuário no programa principal. Após a execução da função, imprima no programa principal o valor retornado pela função.

7) Faça uma função chamada **maior** que recebe como parâmetro dois valores inteiros e retorna qual é o maior número entre dois. Os dois valores devem ser informados pelo usuário no programa principal. Após a execução da função imprima qual é o maior elemento no programa principal.

8) Faça uma função chamada **menor** que recebe como parâmetro um vetor contendo 10 elementos do tipo *float*. A função deve verificar e retornar qual é o menor elemento do vetor. A inicialização do vetor deve ser feita no programa principal e pode usar a função *rand()* para a inicialização do vetor. Após a inicialização do vetor, este deve ser passado como parâmetro para a função. Após a execução da função, imprima no programa principal o menor valor do vetor.

9) Faça 4 funções: a primeira chamada **milhasKm** que recebe como parâmetro um valor de milhas e converte e retorna ao programa principal o valor em quilômetros (uma milha é 1.61 quilômetros). A segunda função chamada **kmMilhas** que recebe como parâmetro um valor que indica os quilômetros. A função deve fazer a conversão de quilômetros em milhas e retornar o valor ao programa principal. A terceira função chamada **pesMetro** recebe como parâmetro um valor que indica a quantidade de pés. A função **pesMetro** deve converter e retornar o valor da conversão ao programa principal (um metro tem 3.28 pés). E a quarta função chamada **milhasPes** recebe como parâmetro um valor que indica as milhas e faz a conversão do valor para pés. A função **milhasPes** deve retornar ao programa principal o valor convertido. Faça a leitura dos valores e as chamadas das quatro funções no programa principal. Imprima os valores retornados pelas funções no programa principal.

10) Faça **uma função** chamada **desvioPadrao** que recebe como parâmetro um vetor de 100 inteiros e retorna o desvio padrão desses valores. O desvio padrão é definido pela raiz quadrada da variância. A variância é definida pelo quadrado da somatória das diferenças entre cada elemento para a média, ou seja, $vr = \sum_{n}^{i=1} (x_i - md)^2$, onde md é a média do conjunto e x_i é cada elemento. Pode ser usada as funções da math.h *pow()* e a função *srqt()*. O vetor já foi inicializado pelo teclado e portanto, não é para fazer a leitura dos dados no vetor.

11) Faça uma **função** chamada **capturaPecaBranca** de um jogo de Damas que recebe como parâmetro uma matriz de inteiros com dimensões 8x8 e dois números

UNIFEI	Universidade Federal de Itajubá Instituto de Engenharia de Sistemas e Tecnologias da Informação-IESTI
Disciplina	ECOP01 – Técnicas de Programação
Professores	Profª.Drª.Thatyana de Faria Piola Seraphim, Prof.Dr.Enzo Seraphim
7ª Lista de Exercícios	

inteiros representando uma linha e coluna qualquer da matriz. A matriz representa a configuração atual do jogo de damas, sendo que, 0 indica uma casa vazia, 1 indica uma casa ocupada por uma peça branca e -1 indica uma casa ocupada por uma peça preta. Para a função retornar verdadeiro deve verificar se existe na matriz uma peça preta na linha e coluna passada para a função e se consegue realizar a captura de uma peça branca. Em qualquer outra situação a função retorna falso. A captura acontece quando a peça preta pula por cima da peça branca estabelecendo-se na posição consecutiva e vazia do tabuleiro. A função faz apenas a verificação se é possível ou não realizar a captura, mas não realiza a captura. Certifique-se que a função não tenha *memory leak*.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	-1		-1		-1		-1	
1		-1		-1		-1		-1
2	-1		-1		-1		-1	
3		0		0		0		0
4	0		0		0		0	
5		1		1		1		1
6	1		1		1		1	
7		1		1		1		1

12) Faça uma função chamada **capturaPecaAzul** de um jogo Reversi que recebe como parâmetro uma matriz de inteiros com dimensões 8x8 e dois números inteiros representando uma linha e coluna qualquer da matriz. A matriz representa a configuração atual do jogo reversi, sendo que, 0 indica uma casa vazia, 1 indica uma casa ocupada por uma peça vermelho e -1 indica uma casa ocupada por uma peça azul. Para a função retornar verdadeiro deve verificar se a posição linha e coluna passada para a função está vazia na matriz e se consegue realizar a captura de peças azuis. Em qualquer outra situação a função retorna falso. Para capturar a peça azul, deve colocar a peça vermelha de forma a encurralar horizontal, vertical ou diagonalmente uma linha de peças azuis consecutivas entre suas peças vermelhas. A função somente faz a verificação se é possível ou não capturar, mas não realiza a captura. Certifique-se que a função não tenha *memory leak*. Abaixo segue um exemplo de captura de peça azul:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	-1	1	0	0	0
4	0	0	0	1	-1	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	1	1	1	0	0	0
4	0	0	0	1	-1	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0