Exercícios 17 – Funções com retorno de valor

17.1 a) Escreva um subalgoritmo chamado **`maiorNumero`** que receba como entrada dois inteiros positivos e retorne o maior deles ou o valor -1 se eles forem iguais. Considere que os valores de entrada são sempre positivos (Não é necessário validar).

Entrada: Dois inteiros positivos.
Retorno: O maior deles ou -1 se eles forem iguais.

b) Escreva um algoritmo para ler um valor N (considere que o valor informado será sempre positivo). A seguir ler N duplas de inteiros. Para cada dupla informada exibir o maior elemento ou a frase **"Eles são iguais"**. Para obter o maior elemento deve ser utilizado o subalgoritmo **`maiorNumero`**.

```
[Entrada]  [Saída]
3 (N)
20  15     20
7   12     12
5   5      Eles são iguais
```

17.2 a) Escreva um subalgoritmo chamado **`ehPrimo`** que receba como entrada um valor inteiro e retorne o valor **1** se ele é primo e **0** caso contrário. Considere que o valor de entrada é sempre positivo.

Entrada: Um valor inteiro.
Retorno: **1** se o inteiro passado como argumento é primo e **0** caso contrário.

b) Reescreva o exercício **11.13** utilizando o subalgoritmo **`ehPrimo`**.

17.3 a) Reescreva o subalgoritmo **`exibePI`** do exercício **16.4** de forma que ao invés de exibir, retorne o valor da série. Utilize o nome **`obtemPI`** para o subalgoritmo.

Entrada: A quantidade de termos utilizada para obter o valor da série.
Retorno: O valor da série.

b) Reescreva o exercício **16.4** b utilizando ao subalgoritmo **obtemPI**.

17.4 a) Escreva um subalgoritmo chamado fatorial que receba como entrada um inteiro não negativo e retorne o seu fatorial.
Exemplo: **`0! = 1`** **`3! = 3x2x1 = 6`**

Entrada: Um inteiro
Retorno: Fatorial do inteiro passado como argumento.

b) Escreva um subalgoritmo chamado **`calculaSerie`** que retorna o valor da série abaixo. A quantidade de termos da série e o valor de x devem ser passados como argumento.

$$S = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \ldots$$

Entrada: A quantidade de termos utilizada para obter o valor da série acima e o valor de x.

c) Escreva um algoritmo para ler várias duplas de dados representando a quantidade de termos da série acima e o valor de x. Para cada valor informado exibir o valor calculado para a série. O programa termina ao ser informado um valor nulo ou negativo para a quantidade de termos (nesta situação o valor de x não deve ser impresso).

```
[Entrada]  [Saída]
1  10       10
2  1        0.833333
3  2        0.933333
5  10       1448.271606
4  5        -5.292658
-1
```

17.5 a) Escreva um subalgoritmo chamado **`somaDivisores`** que receba como entrada um inteiro positivo e retorne a soma de seus divisores (exceto ele mesmo). Considere que o valor de entrada é sempre positivo (Não é necessário validar).

Entrada: Um valor inteiro.
Retorno: A soma de seus divisores (exceto ele mesmo).

b) Escreva um algoritmo para ler vários inteiros. Imprimir para cada um deles a soma de seus divisores (exceto ele mesmo). O

algoritmo termina ao ser informado um valor zero ou negativo. Utilize o subalgoritmo **`somaDivisores`** para obter a soma dos divisores.

```
[Entrada]  [Saída]
10         8
3          1
28         28
1          0
-3
```

17.6 a) Escreva um subalgoritmo chamado **`ehPerfeito`** que receba como entrada um inteiro positivo e retorne **1** se ele for um número perfeito e **0** caso contrário. Utilize o subalgoritmo **`somaDivisores`** para obter a soma dos divisores. Um número é perfeito se ele é igual a soma de seus divisores exceto ele mesmo. Considere que o valor de entrada é sempre positivo.

Entrada: Um valor inteiro.
Retorno: **1** se o número passado como argumento é perfeito e **0** caso contrário.

b) Reescreva o exercício **11.19** utilizando o subalgoritmo **`ehPerfeito`**.

17.7 a) Escreva um subalgoritmo chamado **`saoAmigos`** que receba como entrada dois inteiros positivos e retorne **1** se eles forem números amigos e **0** caso contrário.

OBS: Dois números A e B são amigos se a soma dos divisores de A excluindo A é igual a B e a soma dos divisores de B excluindo B é igual a A. Considere que os dois valores de entrada são positivos. Utilize o subalgoritmo **`somaDivisores`**.

Ex: 220 e 284 são amigos, pois
220: 1+2+4+5+10+11+20+22+44+55+110=284
284: 1+2+4+71+142=220

Entrada: Dois inteiros.
Retorno: **1** se os números passados como argumento são amigos e **0** caso contrário.
b) Escreva um algoritmo para ler um valor N (considere que o valor informado será sempre positivo). A seguir ler N duplas de dados. Para cada dupla informada exibir uma mensagem que indica se os números são ou não amigos. Para verificar se os números são amigos utilize o subalgoritmo **`saoAmigos`**.

```
[Entrada]  [Saída]
3 (N)
20    15     Não são amigos
284   220    São amigos
17296 18416  São amigos
```

17.8 a) Escreva um subalgoritmo chamado **`somaDigitos`** que receba como entrada um inteiro positivo e retorne a soma dos seus dígitos.

`Ex: 1234 ---> 1+2+3+4 = 10 237 ---> 2+3+7 = 12`

Entrada: Um inteiro positivo.
Retorno: A soma dos dígitos do inteiro passado como argumento.

OBS: Considere que o valor de entrada é sempre positivo.

Dica para obter cada dígito: Utilize uma repetição calculando o quociente e o resto da divisão por 10 até que o quociente seja 0. Cada resto obtido equivale a um dígito.

```
237  | 10
     +-----
  7   23 | 10
        +-----
    3    2 | 10
           +-----
      2    0
```

b) Escreva um algoritmo para ler uma quantidade indeterminada de inteiros. Para cada inteiro informado calcular e exibir a soma de seus dígitos. O algoritmo termina ao ser informado um inteiro igual a zero ou negativo. Utilizar o subalgoritmo **`somaDigito`**.

```
[Entrada]  [Saída]
237        12
101        2
1          1
901        10
-3
```

17.9 Para evitar erros de digitação em números de grande importância, como código de uma conta bancária, geralmente se adiciona ao número um dígito verificador. Por exemplo, o número 1841 é utilizado normalmente como 18414, onde o 4 é o dígito verificador. Ele é calculado da seguinte forma:

I) Cada algarismo do número é multiplicado por um peso começando de 2 da direita para a esquerda. Para cada algarismo o peso é acrescido de 1. Soma-se os produtos obtidos.

1x**5** + 8x**4** + 4x**3** + 1x**2** = 51

II) Calcula-se o resto da divisão desta soma por 11:

51 % 11 = 7

III) Subtrai-se de 11 o resto obtido:

11 - 7 = 4

IV) Se o valor obtido for 10 ou 11, o dígito verificador será o 0, nos outros casos, o dígito verificador é o próprio valor encontrado.

a) Escreva um subalgoritmo **`calculaDigito`** que recebe como entrada um número inteiro e retorne o dígito verificador do número conforme descrito acima.

OBS: Considere que o valor de entrada é sempre positivo.

Entrada: Um inteiro.
Retorno: O dígito verificador do número.

b) Escreva um algoritmo para ler vários inteiros. Para cada inteiro informado calcular e exibir o seu respectivo dígito verificador. O algoritmo termina ao ser informado um inteiro igual a zero ou negativo. Utilizar o subalgoritmo **`calculaDigito`**.

```
[Entrada]  [Saída]
1           9
6           0
1841        4
2149        0
2144        0
-3
```

17.10 a) Escreva um subalgoritmo chamado **`numCorreto`** que receba como entrada um inteiro e retorne **1** se o número está correto e **0** caso contrário. Considere que a unidade do número informado representa o seu digito verificador, calculado conforme descrito no exercício anterior. Utilizar o subalgoritmo **`calculaDigito`**.
OBS: Considere que o valor de entrada é sempre positivo.

Entrada: um inteiro.
Retorno: **1** se o número passado como argumento contém um dígito verificador correto e **0** caso contrário.

b) Escreva um algoritmo para ler vários valores inteiros dentro do intervalo de 10 a 999999 onde o último algarismo representa o seu dígito verificador e imprima para cada valor uma mensagem indicando se ele está correto ou não (Ok ou Erro). O programa é encerrado ao ser fornecido um número fora da faixa estabelecida (10 a 999999). Utilize a função **`numCorreto`**.

```
[Entrada]  [Saída]
19          Ok
61          Erro
18414       Ok
21490       Ok
21440       Ok
9
```

17.11 a) Escreva um subalgoritmo chamado **`tam3Nmais1`** que receba como entrada um inteiro e retorne a quantidade de elementos da sequência descrita no exercício **9.10**.
OBS: Considere que o valor de entrada é sempre positivo.

Entrada: um inteiro.
Retorno: a quantidade de elementos da sequência descrita no exercício **9.10**.

b) Escreva um algoritmo para ler um valor N (considere que o valor informado será sempre positivo). A seguir ler N inteiros. Para cada inteiro lido escrever a quantidade de caracteres gerados pela sequência descrita no exercício **9.10**. Utilizar o subalgoritmo **tam3Nmais1**.

```
[Entrada]  [Saída]
4 (N)
2          2
1          1
3          8
16         5
```

c) Escreva um algoritmo para ler um valor N (considere que o valor informado será sempre positivo). A seguir ler N duplas de inteiros que representam um intervalo fechado (considere que serão informados valores positivos). Escrever para cada dupla o tamanho da **maior** sequência gerada pelos números que estão dentro do intervalo. Os elementos da dupla podem estar em qualquer ordem. Utilizar o subalgoritmo **tam3Nmais1**.

Ex: Para 1 e 3 (1 --> tamanho 1, 2--> tamanho 2, 3--> tamanho 8) a resposta é 8.

```
[Entrada]  [Saída]
5 (N)
1 3        8
1 10       20
100 200    125
210 201    89
900 1000   174
```