

# MC102 - Algoritmos e Programação de Computadores

## Lista de Exercícios 5

4 de Outubro de 2016

1. Considere o código em C abaixo (assuma que no lugar de D seja usado o último dígito de seu RA):

```
#include <stdio.h>

int fun1(int a, int b);

int j=1;

int main(){
    int i, j;
    int a = D;///Use o último dígito de seu RA.

    if(a % 2 == 0)
        a = 2;
    else
        a = 3;

    printf("%d\n", fun1(2,4));

    for(i = 1; i<3; i++){
        for(j= 1; j <3; j++){
            printf("%d\n", fun1(a, i+j));
        }
    }
}

int fun1(int a, int b){
    int i, p=1;
    for(i=1; i<=b; i++)
        p = p*a;
    return p+j;
}
```

- (a) **(0.5 pontos)** Determine quais são as variáveis locais e globais deste programa. Para cada variável local identifique a que função esta pertence.
  - (b) **(1.5 pontos)** Mostre o que será impresso na tela do computador quando for executado este programa (lembre-se de usar o último dígito de seu RA no lugar de D).
2. Escreva uma função que recebe dois números inteiros positivos  $a$  e  $b$  por parâmetro e determina se eles são amigos ou não, devolvendo 1 caso sejam amigos, e 0 caso contrário.

Dois números são amigos se cada número é igual à soma dos divisores próprios do outro (os divisores próprios de um número  $m$  são os divisores estritamente menores que  $m$ ). Por exemplo, os divisores próprios de 220 são 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 e 110, cuja soma é 284; e os divisores próprios de 284 são 1, 2, 4, 71 e 142, cuja soma é 220. Logo, 220 e 284 são números amigos. O protótipo da função deve ser:

```
int amigos(int a, int b);
```

3. Escreva uma função que calcule o máximo divisor comum de dois números  $m$  e  $n$  passados por parâmetro. Você deve utilizar a seguinte regra do cálculo do mdc onde  $m \geq n$

$$\text{mdc}(m, n) = m \text{ se } n = 0$$

$$\text{mdc}(m, n) = \text{mdc}(n, m \% n) \text{ se } n > 0$$

O protótipo da função deve ser:

```
int mdc(int m, int n);
```

4. Escreva uma função que recebe um valor inteiro positivo  $n$  como parâmetro e devolve um valor inteiro  $b$  tal que  $b^k = n$  para algum inteiro  $k$ , e  $b$  seja o menor possível. Por exemplo, se  $n = 27$  então o valor devolvido deve ser 3, já se  $n = 12$  então o valor devolvido deve ser 12. Não use funções de bibliotecas na sua solução. O protótipo da função deve ser:

```
int menor_base_log(int n);
```

5. Um inteiro positivo  $n$  é **pitagórico** se existem inteiros positivos  $a$  e  $b$  tais que  $a^2 + b^2 = n$ . Por exemplo, 13 é pitagórico pois  $2^2 + 3^2 = 13$ .

- (a) Escreva uma função que recebe como parâmetro três inteiros  $a$ ,  $b$  e  $n$ , e que devolve 1 caso  $a^2 + b^2 = n$  e devolve 0 caso contrário. O protótipo da função deve ser:

```
int teste(int a, int b, int n);
```

- (b) Utilize a função do item anterior e escreva uma outra função que recebe como parâmetro um inteiro positivo  $n$  e verifica se  $n$  é pitagórico, devolvendo 1 caso  $n$  seja pitagórico e 0 caso contrário. O protótipo da função deve ser:

```
int pitagorico(int n);
```

6. Escreva uma função que recebe um vetor de números reais e o seu tamanho por parâmetro e devolve a média aritmética dos números do vetor. O protótipo da função deve ser:

```
double media(double v[], int tam);
```

7. Escreva uma função que recebe um vetor de números reais e o seu tamanho por parâmetro e devolve o desvio padrão dos números do vetor usando a seguinte fórmula:

$$\sqrt{\frac{1}{n-1} \left( \sum_{i=1}^n x_i^2 - \frac{1}{n} \left( \sum_{i=1}^n x_i \right)^2 \right)}$$

onde  $n$  é o número de elementos. O protótipo da função deve ser:

```
double desvioPadrao(double v[], int tam);
```

8. Escreva uma função que recebe como parâmetro duas matrizes quadradas de tamanho no máximo  $30 \times 30$ , onde a primeira matriz corresponde a uma matriz de entrada e a segunda corresponde a uma matriz resposta. A função deve calcular a transposta da matriz de entrada e guardar o resultado na matriz resposta. A função também recebe como parâmetro um inteiro  $n$  que indica as dimensões das matrizes. O protótipo da função deve ser:

```
void transposta(double mat1[30][30], double matRes[30][30], int n);
```

9. Uma matriz quadrada de inteiros é um quadrado mágico se a soma dos elementos de cada linha, a soma dos elementos de cada coluna, a soma dos elementos da diagonal principal e da diagonal secundária são todos iguais. A matriz abaixo é um exemplo de quadrado mágico:

```
3 4 8
10 5 0
2 6 7
```

Escreva uma função que recebe como parâmetro uma matriz quadrada de tamanho no máximo  $30 \times 30$ , e suas dimensões  $n$ , e determina se ela é um quadrado mágico devolvendo 1 caso seja e 0 caso contrário. O protótipo da função deve ser:

```
int magico(int mat[30][30], int n);
```

10. Escreva uma função que recebe como parâmetro três matrizes quadradas de tamanho no máximo  $30 \times 30$ , onde a primeira e a segunda matriz correspondem a entrada e a terceira corresponde a uma matriz resposta. A função deve calcular a multiplicação da primeira pela segunda matriz e guardar o resultado na matriz resposta. A função também recebe como parâmetro um inteiro  $n$  que indica as dimensões das matrizes. O protótipo da função deve ser:

```
void multiplica(double mat1[30][30], double mat2[30][30], double matRes[30][30], int n);
```

11. Suponha que uma matriz binária represente ligações entre cidades, e que, se uma posição  $(i,j)$  possui o valor 1, então há uma estrada da cidade  $i$  para a cidade  $j$ . Seja o seguinte exemplo de matriz:

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Neste caso, há caminhos disponíveis da cidade 0 para a 1 e 2, e da 2 para 0.

Para cada item abaixo escreva uma função que recebe como parâmetro uma matriz quadrada indicando as estradas entre as cidades, um inteiro  $n$  correspondendo as dimensões da matriz e um vetor resposta (que terá tamanho  $n$ ). O protótipo da função deve ser:

```
void verifica(int mat[30][30], int n, int resposta[]);
```

- A função deve determinar as cidades com entrada e sem saída, indicando isto no vetor resposta, tal que **resposta[i]** recebe 1 caso a cidade  $i$  satisfaça esta propriedade e 0 caso contrário.
- A função deve determinar as cidades com saída mas sem entrada, indicando isto no vetor resposta, tal que **resposta[i]** recebe 1 caso a cidade  $i$  satisfaça esta propriedade e 0 caso contrário.

- A função deve determinar as cidades isoladas, indicando isto no vetor resposta, tal que `resposta[i]` recebe 1 caso a cidade  $i$  satisfaça esta propriedade e 0 caso contrário.

12. No jogo Sudoku temos uma matriz  $9 \times 9$  dividida em 9 quadrados de  $3 \times 3$  preenchidos previamente com alguns números entre 1 e 9 (veja o exemplo à esquerda abaixo). Uma solução para uma instância do jogo consiste no preenchimento de todas as posições vazias com números entre 1 e 9 respeitando-se as seguintes regras:

- Não pode haver números repetidos em um mesmo quadrado, ou seja, cada número entre 1 e 9 deve aparecer exatamente uma vez em cada quadrado.
- Não pode haver números repetidos em nenhuma linha da matriz.
- Não pode haver números repetidos em nenhuma coluna da matriz.

Escreva uma função que recebe uma matriz  $9 \times 9$  por parâmetro que representa uma proposta de solução para um sudoku, e testa se a matriz é ou não uma solução para um sudoku, devolvendo 1 em caso verdadeiro e 0 caso contrário. O protótipo da função deve ser:

```
int solucao(int mat[9][9]);
```

Veja abaixo um exemplo (direita) de uma matriz solução para um sudoku.

	2		5		1			9
8			2		3			6
	3			6				7
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Sudoku não resolvido

4	2	6	5	7	1	3	9	8
8	5	7	2	9	3	1	4	6
1	3	9	4	6	8	2	7	5
9	7	1	3	8	5	6	2	4
5	4	3	7	2	6	8	1	9
6	8	2	1	4	9	7	5	3
7	9	4	6	3	2	5	8	1
2	6	5	8	1	4	9	3	7
3	1	8	9	5	7	4	6	2

Solução