

MAC121 - Algoritmos e Estruturas de Dados I**Segundo semestre de 2020**

“Para fazer uma função recursiva
é preciso ter fé.”
Siang Wun Song

Lista de exercícios – Recursão – entrega: 3 de novembro

IMPORTANTE: Todos os exercícios são interessantes, e devem ser feitos para aprender a matéria.

VOCÊS DEVEM ENTREGAR APENAS OS EXERCÍCIOS 6 e 9

- (a) Faça uma função recursiva MaxMin que calcula o elemento máximo e o elemento mínimo de um vetor com n números inteiros.
(b) Quantas comparações (em função de n) envolvendo elementos do vetor o seu algoritmo faz?
- Considere a função abaixo que calcula o n -ésimo termo da série de Fibonacci:

```
int fibonacci (int n) {  
    printf("*");  
    if ((n == 1) || (n == 2))  
        return (1);  
    return (fibonacci(n-1) + fibonacci(n-2));  
}
```

Quantas “*” são impressas no cálculo de `fibonacci(n)`?

- Faça uma função recursiva Dígito que recebe um número inteiro $n \geq 0$ e conta o número de dígitos de n . Exemplo: se $n = 132$ então $\text{Dígito}(n) = 3$.
- Considere a função abaixo:

```
double f(double x, double y) {  
    if (x >= y)  
        return ((x+y)/2);  
    return (f(f(x+2, y-1), f(x+1, y-2)));  
}
```

Qual é o valor de $f(1, 10)$? Como se poderia calcular $f(a, b)$ de maneira mais simples?

5. A função de Ackermann é definida da seguinte maneira:

$$A(m, n) := \begin{cases} n + 1 & \text{se } m = 0, \\ A(m - 1, 1) & \text{se } m > 0 \text{ e } n = 0, \\ A(m - 1, A(m, n - 1)) & \text{se } m, n > 0. \end{cases}$$

Escreva uma função recursiva que recebe inteiros não negativos m e n e devolve $A(m, n)$.

6. Simule a execução do programa abaixo:

```
int fusc (int n) {
    if (n <= 1) return (1);
    if (n % 2 == 0)
        return (fusc(n / 2));
    return (fusc((n-1)/2) + fusc((n+1)/2));
}

int main() {
    int m = 7;
    printf("Fusc = %d\n", fusc(m));
}
```

7. Considere a seguinte função:

```
void misterio (int A[], int inic, int fim) {
    int aux;
    while (A[fim] % 2 == 0 && inic < fim)
        fim--;
    while (A[inic] % 2 == 1 && inic < fim)
        inic++;
    if (inic < fim){
        aux = A[inic];
        A[inic] = A[fim];
        A[fim] = aux;
        misterio(A, inic, fim);
    }
}
```

(a) Simule a função mistério para

$A =$

8	10	3	6	5	2	9	1	4
---	----	---	---	---	---	---	---	---

 Início= 0 e Fim = 8.

(b) O que faz a função mistério? Quantas comparações envolvendo elementos do vetor A são feitas? Escreva um algoritmo que faz a mesma coisa com um número menor de comparações.

8. Simule a seguinte função recursiva para $n = 6$:

```

int zzz(int n) {
    int aux;
    if (n <= 2)
        return(1);
    n--;
    aux = zzz(n);
    n--;
    return (aux + zzz(n));
}

```

O que faz a função `zzz`?

9. Escreva uma função recursiva de protótipo

```
par tabela (int n);
```

onde,

```

typedef struct {
    int x;
    int y;
} par;

```

que recebe como parâmetro um inteiro não negativo n e calcula um par de inteiros (x, y) , onde x e y são as coordenadas de n na tabela abaixo:

	0	1	2	3	4	...	Y
0	0	2	5	9	14		
1	1	4	8	13			
2	3	7	12				
3	6	11					
4	10						
...							
X							N

10. A função abaixo calcula o maior divisor comum dos inteiros positivos m e n . Escreva uma função recursiva equivalente.

```

int Euclides (int m, int n) {
    int r=m%n;
    while (r != 0){
        m = n;
        n = r;
        r = m % n;
    }
    return(n);
}

```

11. Considere o problema 10 da lista de pilhas, sobre o problema das 4 cores. Faça uma função recursiva com protótipo:

```
int quatrocores(int **mapa, int n, int *cores, int atual)
```

que recebe uma matriz de vizinhanças representando um mapa, o número de países, um vetor parcialmente preenchido com as cores dos países e o índice de um país atual e devolve 1 se é possível colorir os países com 4 cores de forma que países vizinhos tenham cores diferentes, e zero caso contrário.