Lista de exercícios--Recursão

``Para fazer uma função recursiva é preciso ter fé."
Siang Wun Song apud Ricardo Luís Lachi

1. Resolva os itens a seguir:
   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?

2. Faça uma função recursiva **Dígito** que recebe um número inteiro **n** e calcula a soma dos dígitos de **n**. Exemplo: se **n = 132** então **Dígito(n) = 6**.

* 3. Dada uma string de comprimento **n** do tipo: S = "(4 * 3 + (3 - 2))"
   a. Faça uma função recursiva que verifica se uma expressão de parênteses é bem formada.
   b. Idem para uma expressão com parênteses, colchetes (`[',`]') e chaves (`{',`}').

4. 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[1] 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)**.

[1] Mais informações sobre a função de Akermann procure na biblioteca o livro: "Cormen, Thomas H. **Algorítmos: teoria e prática.** 2002".

6. Simule a execução do programa abaixo:

```
#include <stdio.h>
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

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
A	8	10	3	6	5	2	9	1	4	**Inicio=0 e Fim=8**

   b. O que faz a função Mistério? Quantas comparações envolvendo elementos do vetor **A** são feitas? Escreva um algoritmo e implemente um programa desse 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));
}
```

* 9. Escreva uma função recursiva **Tab(N,X,Y)** 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. Suponha que temos de calcular o produto de matrizes

$$\mathbf{M_1 \times M_2 \times \ldots \times M_n}$$

onde cada $\mathbf{M_i}$ é uma matriz com $\mathbf{r_i}$ linhas e $\mathbf{r_{i+1}}$ colunas (portanto, $\mathbf{r_1,\ldots,r_{n+1}}$ descrevem as dimensões das matrizes). Suponha ainda que o produto de qualquer par de matrizes será calculado pelo algoritmo usual; assim o produto de uma matriz **p x q** por uma matriz **q x r** requer **p x q x r** operações de multiplicação entre números. A ordem em que a multiplicação de três ou mais matrizes é executada pode afetar sensivelmente o número total de multiplicações. Por exemplo, se **n = 4** e $\mathbf{(r_1,\ldots,r_5) = (10, 20, 50, 1, 100)}$ então o cálculo da expressão $\mathbf{M_1 \times (M_2 \times (M_3 \times M_4))}$ requer 125000 multiplicações enquanto que o cálculo da expressão $\mathbf{(M_1 \times (M_2 \times M_3)) \times M_4}$ requer 2200 multiplicações. Seja $\mathbf{m_{i,j}}$ o número mínimo de multiplicações necessárias para calcular $\mathbf{M_i \times \ldots \times M_j}$. Temos que

$$m_{i,j} = \begin{cases} 0 & \text{se } i = j \\ \min_{i \le k \le j} \{ m_{i,k} + m_{k+1,j} + r_i \times r_{k+1} \times r_{j+1} \} & \text{se } i < j \end{cases}$$

Escreva uma função recursiva que recebe **n** e a seqüência **(r_1, ..., r_{n+1})** e calcula **$m_{1,n}$**.

* 11. Escreva uma função que dado **n** imprime todas as permutações dos números inteiros **1,...,n**. Escreva duas versões, uma iterativa e uma recursiva. (Sugestão: O conjunto das permutações dos inteiros **1,...,n** pode ser obtida através do conjunto das permutações dos inteiros de **1,...,n-1** inserindo-se **n** em cada possível posição de cada permutação.)

* 12. Escreva uma função que dados dois inteiros positivos **n** e **k** imprima todas as combinações de **1,...,n** em grupos de tamanho **k**. Escreva duas versões, uma iterativa e outra recursiva.

* 13. Escreva um programa para imprimir em ordem lexicográfica todos os subconjuntos do conjunto **{1,...,n}**. Para **n = 4** o resultado deve ser:

Ø {1} {1,2}{1,2,3}{1,2,3,4}{1,2,4}{1,3}{1,3,4}
{1,4}{2}{2,3}{2,3,4}{2,4}{3}{3,4}{4}

14. 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;
  do{
     r = m % n;
     m = n;
     n = r;
  }
  while (r != 0);
  return(m);
}
```