EXERCÍCIOS DE RECURSIVIDADE

1) O que a seguinte função faz?

```
public int misterio(int a, int b){

      if (b == 1) return a;

      else return a + misterio(a, b-1);

}
```

2) Localize o(s) erro(s) na seguinte função recursiva e explique como corrigi-lo(s). Essa função deve calcular a soma dos valores de 0 a n.

```
public int soma(int n){

      if (n == 0) return 0;

      else return n + soma(n);

}
```

3) Escreva uma função recursiva

potencia(base,expoente)

que, quando chamada, retorna $base^{expoente}$. Por exemplo, potencia(3,4) = 3*3*3*3. Assuma que expoente é um inteiro maior ou igual a 0. Dica: o passo de recursão deve utilizar o relacionamaento

$$base^{expoente} = base * base^{expoente-1}$$

e a condição de terminação ocorre quando expoente é igual a 0, porque

$$base^{0} = 1$$

Incorpore esse método em um progrma que permita que o usuário informe a base e o expoente da potenciação.

4) Escreva duas definições para uma função soma que, dados dois números inteiros não negativos a e b, retorne a sua

soma a + b, usando apenas as operações mais simples de incrementar 1 e decrementar 1 (suponha que as operações de adicionar e de subtrair mais de uma unidade não estão disponíveis). A primeira definição deve usar um comando de repetição, e a segunda definição deve ser recursiva.

5) Qual o resultado da execução do programa abaixo?

```
public class Teste{

      public int ff (int n){

        if (n == 1) return 1;

        if (n % 2 == 0) return ff(n/2);

        return ff((n-1)/2) + ff((n+1)/2);

     }

      public static void main (String args[]){

        System.out.println(ff (7));

      }

}
```

6) Determine a saída do seguinte programa, quando executado.

```
public class Teste{

      public int fusc(int n, int profund){

        int i;

        for (i = 0; i < profund; i++)

         System.out.print("...");

        System.out.println("fusc(" + n + ", " + profund + ")");

        if (n == 1) return 1;

        if (n % 2 == 0) return fusc(n/2, profund+1);

        return fusc((n-1)/2, profund+1) + fusc((n+1)/2, profund+1);

      }

      public static void main(String args[]){

       fusc(7,0);

      }

}
```

7) Implemente e teste a função h definida recursivamente por:

$$h(m,n) = \begin{cases} m+1, & se\ n = 1 \\ n+1, & se\ m = 1 \\ h(m,n-1) + h(m-1,n), & se\ m > 1\ e\ n > 1 \end{cases}$$

8) O máximo divisor comum (MDC) de dois números inteiros x e y pode ser calculado usando-se uma definição recursiva:

MDC(x, y) = MDC(x - y, y), se x > y .

Além disso, sabe-se que:

MDC(x, x) = x

MDC(x, y) = MDC(y, x)

Exemplo:

MDC(10,6)= MDC(4,6)= MDC(6,4)= MDC(2,4)= MDC(4,2)= MDC(2,2)= 2

Então, pede-se que seja criada uma função recursiva para descrever tal definição. Crie, também, um algoritmo que leia os dois valores inteiros e utilize a função criada para calcular o MDC de x e y, e imprima o valor computado.

9) Escreva uma função recursiva que calcule *nx* onde *n* e *x* são valores inteiros.

10) Escreva uma função recursiva que calcule a soma dos n valores inteiros ímpares.

11) Escreva uma função recursiva que calcule o resto (MOD) da divisão entre dois números inteiros n e m.

Sejam as regras:

MOD(n, m) = MOD(n-m, m), se n>=m.

MOD(n, m) = n, se n<m.

12) Escreva um método recursivo que converta um valor passado como parâmetro de Decimal para Hexadecimal.

13) Escreva um método recursivo que calcule a seguinte função:

F(x) = x*x, se x é múltiplo de 3.

F(x) = x+3, se o resto da divisão de x por 3 for 1

F(x) = x!, se o resto da divisão de x por 3 for 2

14) Usando a função puzzle dada abaixo, responda os itens a, b e c.

```
public int puzzle(int base, int limit)

{      //base and limit are nonnegative numbers

       if ( base > limit ) return -1;

       else if ( base == limit ) return 1;

       else return base * puzzle(base + 1, limit);

}
```

a) Identifique a(s) solução(ões) triviais de puzzle.
b) Identifique a chamada recursiva de puzzle e quando ela ocorre.
c) Apresente o resultado para as seguintes rodadas de puzzle:
   a. puzzle(14,10)
   b. puzzle(4,7)
   c. puzzle(0,0)

15) Complete o código java para calcular recursivamente a seguinte expressão: sum = 1 + 1/2 + 1/3 +...+1/n, n>1.

```
public double sum(int n)          // n>=1
{
       if (____________)
              return _______________;
       return _______ + sum(_______);
}
```

16) Escreva um método recursivo que calcule a seguinte somatória:

$$S(N) = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + \ldots +/- 1/N$$

17) Dada a implementação da função abaixo:

```
public int F(int N){
       if (N < 4) return 3*N
       else return 2 * F(N - 4) + 5
}
```

Quais são os valores de F(3) e de F(7)?

18) Algoritmo de ordenação por inserção: Este algoritmo se baseia na seguinte estratégia:

Localize o menor valor do vetor e troque com o valor da primeira posição.

Na sequencia, localize o segundo menor valor do vetor e troque com a segunda posição.
E o processo continua até existam somente as duas posições finais do vetor a se ordenar. Neste caso, compara-se os dois valores e realiza-se a troca se estiverem fora de ordem.

Pode-se utilizar recursividade para resolver este problema usando a seguinte idéia:
Para ordenar o vetor, basta localizar o menor de todos os valores e trocar com a primeira posição do vetor.
A partir daí é só ordenar o restante do vetor (a partir da segunda posição) – Chamada recursiva.
Quando o vetor apresentar duas posições, deve-se compará-las e trocá-las caso estejam fora de ordem – Solução trivial.

Construir o método que realiza a ordenação de um vetor utilizando a estratégia do algoritmo de ordenação por inserção e aplique esse algoritmo com recursividade.

19) A recursividade pode ser utilizada para gerar todas as possíveis permutações de um conjunto de símbolos. Por exemplo, existem seis permutações no conjunto de símbolos A, B e C: ABC, ACB, BAC, BCA, CBA e CAB. O conjunto de permutações de N símbolos é gerado tomando-se cada símbolo por vez e prefixando-o a todas as permutações que resultam dos (N - 1) símbolos restantes. Consequentemente, permutações num conjunto de símbolos podem ser especificadas em termos de permutações num conjunto menor de símbolos. Formule um algoritmo recursivo para este problema.
20) Considere uma partida de futebol entre duas equipes A x B, cujo placar final é M x N, em que M e N são os números de gols marcados por A e B, respectivamente. Implemente um algoritmo recursivo que imprima todas as possíveis sucessões de gols marcados. Por exemplo, para um resultado de 3 x 1 as possíveis sucessões de gols são "A A A B”, "A A B A”, "A B A A" e "B A A A".
21) Uma estratégia que poderia ser utilizada para determinar o maior valor de um vetor consiste em considerar que o maior valor é o valor armazenado na primeira posição ou o maior valor existente no restante do vetor (a partir da segunda posição). Isso leva a usar recursividade uma vez que para o maior valor do vetor restante, pode-se utilizar a mesma estratégia. Agora, quando o vetor que se busca determinar o maior valor possui somente uma posição, o valor armazenado nessa posição é o maior. Construa um método recursivo que retorne o maior valor armazenado em um vetor.