Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Estruturas de Dados I – BCC202
Professor: David Menotti (menottid@gmail.com)
Professora: Emiliana Mara Lopes Simões (simoes.eml@gmail.com)

Lista de Exercícios sobre Recursividade

1) Dado os algoritmos recursivos abaixo, apresente suas funções de complexidade de tempo.

```
a)
void Pesquisa(int n)
{
  if (n > 1)
  {
    Inspecione n*n*n elementos; // custo n^3
    Pesquisa (n/3);
  }
}

b)
void Pesquisa(n)
{
  if ( n <= 1 )
    return;
  else
  {
    obtenha o maior elemento dentre os n elementos;
    /* de alguma forma isto permite descartar 2/5 dos*/
    /* elementos e fazer uma chamada recursiva no resto*/
    Pesquisa(3n/5);
  }
}

c)
void Pesquisa(int A[n], int n);
{
  if ( n <= 1 )
    return;
  else
  {
    ordena os n elementos;
    /* de alguma forma isto permite descartar 1/3 dos */
    /* elementos e fazer uma chamada recursiva no resto*/
    Pesquisa (2n/3);
  }
}
```

d)

```
int fib(int n)
{
  if ( n == 0 )
    return 0;
  else if (n == 1)
    return 1;
  else
    return Fib(n-1) + Fib(n-2);
}
```

e)

```
void  Proc( int n )
{
  if ( n == 0)
    return 1;
  else
    return Proc(n-1) + Proc(n-1);
}
```

f)

```
/* n é uma potencia de 2 */
void Sort (int A[n],int i, int j)
{
  if ( i < j )
  {
    m = (i + j - 1)/2;

    Sort(A,i,m);     /* custo = T(N/2) */
    Sort(A,m+1,j);   /* custo = T(N/2) */
    Merge(A,i,m,j); /* custo = N-1 comparacoes no */
    /* pior caso */
    /* Merge intercala A[i..m] e A[m+1..j] em A[i..j] */
  }
}
```

```
g)
/* n uma potencia de 3 */
void Sort2 (int A[n], int i, int j)
{
  if ( i < j )
  {
    m = ( (j - i) + 1 )/3;
    Sort2(A,i,i+m-1);
    Sort2(A,i+m,i+ 2m -1);
    Sort2(A,i+2m,j);
    Merge(A,i,i+m,i+2m, j);
    /* Merge intercala A[i..(i+m-1)], A[(i+m)..(i+2m-1) e
A[i+2m..j] em A[i..j] a um custo ( (5n/3) -2 ) */
  }
}
```

2) Implemente uma função recursiva que, dados dois números inteiros x e n, calcula o valor de x^n. Escreva e resolva a equação de recorrência dessa função. Qual é a ordem de complexidade da sua função? Qual seria a ordem de complexidade dessa mesma função implementada sem utilizar recursividade? O que você conclui?

3) Considere a função abaixo:

```
int X(int a)
{
  if ( a <= 0 )
    return 0;
  else
    return a + X(a-1);
}
```

   a. O que essa função faz?
   b. Calcule a sua ordem de complexidade. Mostre como você chegou a esse resultado.
   c. Escreva uma função não-recursiva que resolve o mesmo problema. Qual é a ordem de complexidade da sua função? Explique.
   d. Qual implementação é mais eficiente? Justifique.

4) Vários algoritmos em computação usam a técnica de "Dividir para Conquistar": basicamente eles fazem alguma operação sobre todos os dados, e depois dividem o problema em sub-problemas menores, repetindo a operação. Uma equação de recorrência típica para esse tipo de algoritmo é mostrada abaixo. Resolva essa equação de recorrência.

```
T(n) = 2T(n/2) + n;
T(1) = 1;
```

5) Um problema típico em ciência da computação consiste em converter um número da sua forma decimal para a forma binária. Por exemplo, o número 12 tem a sua representação binária igual a 1100. A forma mais simples de fazer isso é dividir o número sucessivamente por 2, onde o resto da *i-ésima* divisão vai ser o dígito *i* do número binário (da direita para a esquerda).

   Por exemplo: 12 / 2 = 6, resto **0** (1º dígito da direita para esquerda), 6 / 2 = 3, resto **0** (2º dígito da direita para esquerda), 3 / 2 = 1 resto **1** (3º dígito da direita para esquerda), 1 / 2 = 0 resto **1** (4º dígito da direita para esquerda). Resultado: **12 = 1100**

   a. Escreva um procedimento recursivo `Dec2Bin(n: integer)` que dado um número decimal imprima a sua representação binária corretamente.

   b. Calcule qual é a ordem de complexidade do seu procedimento. Para isso, **determine e resolva** a equação de recorrência desse procedimento recursivo.

   c. Utilizando um dos tipos abstratos de dados vistos em sala, implemente um procedimento que faça a mesma coisa, ou seja, dado um número decimal imprima a sua representação binária.

6) Considere um sistema numérico que não tenha a operação de adição implementada e que vc disponha somente dos operadores (funções) sucessor e predecessor. Então, pede-se para escrever uma função recursiva que calcule a soma de dois números x e y através desses dois operadores: sucessor e predecessor.

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

$$MDC(x, y) = MDC(x - y, y), \text{ se } x > y .$$

   Além disso, sabe-se que:

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

   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.

8) Pode-se calcular o resto da divisão, ***MOD***, de ***x*** por ***y***, dois números inteiros, usando-se a seguinte definição:

$$MOD(x,y) = \begin{cases} MOD(|x|-|y|,|y|), & \text{se } |x| > |y| \\ |x| & \text{se } |x| < |y| \\ 0 & \text{se } |x| = |y| \end{cases}$$

Então, pede-se que seja criada uma função recursiva para descrever tal definição. A função deve retornar -1 caso não seja possível realizar o cálculo. Além disso, crie um algoritmo que leia os dois valores inteiros e utilize a função criada para calcular o resto da divisão de ***x*** por ***y***, e imprima o valor computado.

9) Pode-se calcular o quociente da divisão, ***DIV***, de ***x*** por ***y***, dois números inteiros, usando-se a seguinte definição:

$$DIV(x,y) = \begin{cases} 1 + DIV(|x|-|y|,|y|), & \text{se } |x| > |y| \\ 0 & \text{se } |x| < |y| \\ 1 & \text{se } |x| = |y| \end{cases}$$

Então, pede-se que seja criada uma função recursiva para descrever tal definição. A função deve retornar -1 caso não seja possível realizar o cálculo. Além disso, crie um algoritmo que leia os dois valores inteiros e utilize a função criada para calcular o quociente de ***x*** por ***y***, e imprima o valor computado.

10) Seja a série de Fibonacci:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

que pode ser definida recursivamente por:

$$Fib(n) = \begin{cases} 1 & \text{se } n = 1 \vee n = 2 \\ Fib(n-1) + Fib(n-2) & \text{se } n > 2 \end{cases}$$

Então escreva:

- Uma função recursiva que gere o termo de ordem n da série de Fibonacci.
- Um algoritmo que, utilizando a função definida acima gere a série de Fibonacci até o termo de ordem 20.

11) O mínimo múltiplo comum (***M.M.C.***) entre dois números inteiros e positivos ***X*** e ***Y*** é definido como sendo o menor inteiro positivo, que seja múltiplo comum a ***X*** e ***Y***. Pede-se que seja criada uma função recursiva (não serão aceitas funções não recursivas) para o cálculo do ***M.M.C.***, onde a função deverá retornar 0 caso não seja possível computar o ***M.M.C.*** e o valor do ***M.M.C.*** entre ***X*** e ***Y*** em caso contrário. Então, apresenta-se a seguinte definição recursiva que deve ser implementada:

$$M.M.C.(X,Y) = \begin{cases} Z * M.M.C.(X/Z, Y/Z), & \text{se } X \bmod Z = 0 \text{ e } Y \bmod Z = 0 \text{ para } 1 < Z \le X,Y \\ Z * M.M.C.(X/Z, Y) & \text{se } X \bmod Z = 0 \text{ e } Y \bmod Z \neq 0 \text{ para } 1 < Z \le X,Y \\ Z * M.M.C.(X, Y/Z) & \text{se } X \bmod Z \neq 0 \text{ e } Y \bmod Z = 0 \text{ para } 1 < Z \le X,Y \end{cases}$$

$$M.M.C.(1,1) = 1$$

Escreva também um algoritmo para testar a função criada.

12) Implemente uma função recursiva `soma(n)` que calcula o somatório dos n primeiros números inteiros. Escreva e resolva a equação de recorrência dessa função. Qual é a ordem de complexidade da sua função? Qual seria a ordem de complexidade dessa mesma função implementada sem utilizar recursividade? O que você conclui?