Lista de exercícios sobre recursão
Prof. João B. Oliveira

1. Escreva um método recursivo `fatorial(int n)` que calcula o fatorial de `n` e produz **exatamente** a saída abaixo para o caso `n`=5:

```
Fatorial de 5
   Fatorial de 4
      Fatorial de 3
         Fatorial de 2
            Fatorial de 1
               Fatorial de 0
               Retorna 1
            Retorna 1
         Retorna 2
      Retorna 6
   Retorna 24
Retorna 120
```

2. Escreva um método recursivo `soma(int i, int j)` que recebe `i` e `j` e retorna a soma de todos os números de `i` até `j`, **mas faz isso dividindo o trecho em duas partes e depois soma cada parte separadamente.**

3. Escreva um método recursivo `pow(int x, int n)` que recebe `x` e `n` e calcula x^n. Seu método também deve funcionar corretamente quando $n = 0$ ou $n = 1$.

4. Adapte as ideias usadas no método anterior e escreva um método recursivo `mult(int i, int j)` que recebe dois inteiros `i` e `j` e retorna $i * j$ mas dentro do seu método não pode haver multiplicações, apenas somas.

5. Escreva um método recursivo `boolean pal(char [] txt)` que recebe `txt` e determina se `txt` é um palíndromo. Faça dois métodos diferentes, cada um explorando um tipo de recursão:

   (a) Usando um método recursivo auxiliar que recebe `txt` e posições `i` e `j` para testar;

   (b) Sem usar um método auxiliar, mas criando um novo `txt` menor e usando recursão.

6. Escreva um método recursivo `numerodeuns(int n)` que recebe um número `n` e retorna quantos digitos 1 existem na representação binária de `n`.

7. Escreva um método recursivo `rev(char [] txt)` que recebe `txt` e reverte as palavras dentro de `txt`, mas mantendo cada palavra no seu lugar original, apenas invertida. Por exemplo, "hello world" se transforma em "olleh dlrow".

8. Escreva um método recursivo `pow2(int x, int n)` que recebe `x` e `n` e calcula x^n usando as regras abaixo:

   (a) Se n é par, então $x^n = x^{n/2} * x^{n/2}$

   (b) Se n é ímpar, então $x^n = x * x^{n/2} * x^{n/2}$

   Depois de fazer o método, coloque um contador e conte quantas multiplicações são feitas por ele. Verifique se é mais econômico do que o método `pow(int x, int n)`. Descubra como otimizar o método para que ele seja mais eficiente.

9. Escreva um programa que recebe um número $x < 1$ e apresenta uma soma de frações que aproxima x. Por exemplo, se recebermos $x = 0.543$ uma soma de frações que pode aproximar x seria

$$x \approx 1/2 + 1/24 + 1/750 = 0.5429999999999$$

Seu programa deve ter algum tipo de controle para o grau de aproximação desejado pelo usuário.

10. Descubra o que faz o método abaixo, que é chamado pela primeira vez como `m(V, 0, V.length-1)` onde `V` é um vetor de inteiros:

```
m(int[] V, int i, int j) {
  if ( i == j ) { print V[i]; return; }
  int mm = (i+j)/2;
  m(V, i, mm);
  m(V, mm+1, j);
}
```

11. Escreva um método **não-recursivo** `int[] combinavetores(int[] v1, int[] v2)` que recebe os vetores `v1` e `v2` (que estão ordenados) e combina os dois criando um novo vetor ordenado contendo todos os elementos de `v1` e de `v2`.

12. Use o método acima para escrever um método recursivo que ordena um vetor. (Se você não souber como pode fazer isso, investigue o que é o **mergesort**)

13. Experimente o algoritmo abaixo com a string "`1+2-3*4`" e outras similares. Descubra o que ele faz e ache alguns dos seus problemas. No algoritmo, `s(i,j)` representa a substring de `s` que vai do caractere `i` até o caractere `j`.

```
func eval(String s) :
  para i de 1 a length(s) :
    switch s[i]:
      case "+" : return eval(s(1,i)) + eval(s(i+1,length(s))
      case "-" : return eval(s(1,i)) - eval(s(i+1,length(s))
      case "*" : return eval(s(1,i)) * eval(s(i+1,length(s))
      case "/" : return eval(s(1,i)) / eval(s(i+1,length(s))
  return toInt(s)
```

14. Em uma terra distante os sacerdotes construíram uma grande plataforma onde foram empilhados muitos discos de ouro, cada um um pouco menor do que o outro e formando algo parecido com uma pirâmide circular. Infelizmente, logo depois que a plataforma ficou pronta eles foram visitados por seres em uma nave alienígena, que os ameaçaram:

> – *Vamos construir mais duas plataformas pra vocês, mas sem os discos de ouro. As três serão chamadas de Plim, Blorg e Snurf, em homenagem a nossos deuses. Caso contrário vocês serão destruídos.*
>
> – Hmmm, certo, ótimos nomes...
>
> – *E vocês tem a missão de transportar todos os discos da plataforma Plim para Snurf, mas só podem mover um de cada vez...*
>
> – Mas...
>
> – *... caso contrário serão destruídos. Vocês só podem colocar discos pequenos em cima de maiores, nunca o contrário, senão...*
>
> – A gente já entendeu, pula essa parte...
>
> – *E vocês também podem usar a terceira torre como depósito. Mas nada de empilhar grandes em pequenos!*

Depois de experimentar um pouco, os sacerdotes descobriram que se tivessem apenas um disco, fazer a mudança seria fácil:

`Mova o disco que está em` *`Plim`* `para` *`Snurf`*

E se tivessem uma pilha de dois discos também seria simples:

`Mova o disco que está em` *`Plim`* `para` *`Blorg`*

`Mova o disco que está em` *`Plim`* `para` *`Snurf`*

`Mova o disco que está em` *`Blorg`* `para` *`Snurf`*

(Faça um desenho e confirme que funciona.) Agora você deve ajudar os sacerdotes escrevendo um algoritmo que encontra a sequência de movimentos para mover um número qualquer de discos entre as torres.