PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Outros Exercícios sobre Recursividade

Curso : *Engenharia de Software*
Disciplina : *Algoritmos e Estruturas de Dados II*
Professora : *Eveline Alonso Veloso*

Regras Básicas:

1. Estude bastante cada par de entrada/saída.
2. Todos os programas deverão ser desenvolvidos na linguagem de programação Java.
3. Estes exercícios deverão ser desenvolvidos individualmente.
4. Cópias de exercício, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
5. Nestes exercícios, vocês não poderão usar os métodos da classe *String*, a não ser os métodos: *char charAt*(*int*), *boolean equals(String)* e *int length*(). Outras classes de Java que manipulam *strings* não são permitidas nesse momento.
6. Para cada exercício, vocês devem submeter apenas um arquivo (.java). Essa regra será necessária para a submissão de trabalhos no VERDE.
7. A resolução (código) de cada exercício deverá ser submetida ao VERDE.
8. A correção será realizada automaticamente pelo VERDE e validada por meio de apresentações durante as aulas práticas da disciplina.

Dicas sobre Recursividade:

1. Um método recursivo pode ser utilizado como uma estrutura de repetição e um erro básico ao se utilizar recursividade consiste em confundir tais estruturas com os métodos recursivos. Para evitar esse erro, nestes exercícios, vocês não poderão usar estruturas de repetição nos métodos recursivos.
2. Outro erro comum consiste em criar variáveis globais (em Java, atributos) para darem suporte à recursividade. Para evitar esse erro, nestes exercícios, vocês não poderão usar variáveis globais.
3. Um terceiro erro comum consiste em passar o valor de retorno como parâmetro para o método recursivo. Para evitar esse erro, nestes exercícios, vocês não poderão passar parâmetros extras aos do enunciado, exceto um contador para a recursividade.
   O método abaixo apresenta um exemplo desse erro.

```
int contarLetrasMinusculas(String s , int i , int resp){

        int contar;

        if (i == s.length())
```

```
            contar = resp;
        else if (s.charAt(i) >= 'a' && s.charAt(i) <= 'z')
            contar = contarLetrasMinusculas (s, i + 1, resp + 1);
        else
            contar = contarLetrasMinusculas ( s , i + 1 , resp) ;

        return contar ;
}
```

4. Quando criamos um método recursivo, normalmente, passamos um contador como parâmetro. Uma forma elegante de implementar essa solução recursiva consiste em criar dois métodos: um recursivo, privado; e outro, público, que chama o método recursivo pela primeira vez e inicializa o valor do contador, encapsulando, dessa maneira, a utilização da recursividade.
   Nestes exercícios, devemos usar essa técnica em todos os métodos recursivos. Veja o exemplo abaixo:

```
public boolean somenteLetrasMinusculas(String s) {
        return somenteLetrasMinusculas (s, 0);
}

private boolean somenteLetrasMinusculas(String s, int i) {

        boolean resp ;

        if (i == s.length())
            resp = true;
        else if ((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') == false)
            resp = false;
        else
            resp = somenteLetrasMinusculas (s, i + 1) ;

        return resp;
}
```

5. Uma boa prática de programação consiste em empregar um único *return* em cada método. Por exemplo, o método anterior é mais indicado que a versão abaixo:

```
public boolean somenteLetrasMinusculas(String s) {
        return somenteLetrasMinusculas (s, 0);
}

private boolean somenteLetrasMinusculas(String s, int i) {

        if (i == s.length())
```

```
            return true;
        else if ((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') == false)
            return false;
        else
            return somenteLetrasMinusculas (s, i + 1) ;
}
```

Exercícios:

1. **Is recursivo**
   Crie um método recursivo que receba uma *string* e retorne *true* se a mesma for composta somente por vogais. Crie outro método recursivo que receba uma *string* e retorne *true* se a mesma for composta somente por consoantes. Crie um terceiro método recursivo que receba uma *string* e retorne *true* se a mesma corresponder a um número inteiro. Crie um quarto método recursivo que receba uma *string* e retorne *true* se a mesma corresponder a um número real.
   Na saída padrão, para cada linha de entrada, escreva outra de saída da seguinte forma X1 X2 X3 X4 onde cada Xi é um booleano indicando se a entrada é: composta somente por vogais (X1); composta somente por consoantes (X2); um número inteiro (X3); um número real (X4). Se Xi for verdadeiro, seu valor deverá ser SIM; caso contrário, NAO.

2. **Álgebra booleana**
   Crie um método recursivo que receba uma *string* contendo uma expressão booleana e os valores de suas entradas e retorne um número inteiro indicando se a expressão é verdadeira ou falsa. Cada *string* da entrada padrão é composta por um número inteiro *n*, que indica o número de entradas da expressão booleana corrente. Em seguida, a string contém *n* valores binários (um para cada entrada) e a expressão booleana. Na saída padrão, para cada linha de entrada, escreva uma linha de saída com 1 ou 0, indicando se a expressão corrente é verdadeira ou falsa.