Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação II
Gabarito da AD1 - 1° semestre de 2015

Questão 1 (2,5 pontos)
Construa expressões regulares em PHP para validar as sintaxes das seguintes entradas de dados (escreva apenas as expressões):

a. Código postal (CEP)

```
R:     ^[0-9]{5}-[0-9]{3}$
```

b. Letras em uma sequência de DNA de qualquer tamanho

```
R:     ^(A|C|G|T)+$
```

c. Endereço na internet. A expressão regular deve aceitar domínios sem o prefixo "www", além de tratar o caso de sites seguros por certificado (https). Exemplos:

```
http://www.domínio.com
https://www.domínio.com
http://domínio.com
http://domínio.com.br
http://www.domínio.qualquer.subdominio.edu.jp
```

```
R:
     ^(http\:\/\/|https\:\/\/)([A-zÀ-ú0-9][A-zÀ-ú-9\-]*\.)+[A-zÀ
-ú0-9][A-zÀ-ú0-9\-]*$
```

d. Placas de automóveis nacionais. Exemplos:

```
ADF 1234
KXE 4567
DEF 0988
```

```
R:     ^[A-Z]{3} \d{4}$
```

Questão 2 (1,5 pontos)
Suponha um vetor armazenando dados de alunos associando a matrícula ao nome do aluno, como por exemplo:

```
$alunos = array(
    12345 => "João da Silva",
    12567 => "Maria Almeida",
    12677 => "Bernardo Nóbrega",
    13567 => "Arnaldo Albuquerque"
);
```

Escreva uma função PHP que recebe o vetor de alunos desordenado e ordena o mesmo pelo número de matrícula usando recursão. Sua função deve subdividir o vetor em duas metades durante a recursão. Considere uma maneira eficiente de dividir o vetor durante o processo de ordenação recursiva.

R:

```
<?php

function mergesort($arr)
{
    if (count($arr) == 1)
        return $arr;
    $mid   = count($arr) / 2;
    $left  = array_slice($arr, 0, $mid, true);
    $right = array_slice($arr, $mid, count($arr), true);
    $left  = mergesort($left);
    $right = mergesort($right);
    return merge($left, $right);
}

function merge($left, $right)
{
    $res = array();
    while (count($left) > 0 && count($right) > 0) {

        $kl = key($left);
        $kr = key($right);

        $vl = current($left);
        $vr = current($right);

        if ($kl > $kr) {
            $res[$kr] = $vr;
```

```
            $right    = array_slice($right, 1, count($right),
true);
        } else {
            $res[$kl] = $vl;
            $left     = array_slice($left, 1, count($left),
true);
        }
    }
    while (count($left) > 0) {
        $res[$kl] = $vl;
        $left     = array_slice($left, 1, count($left), true);
    }
    while (count($right) > 0) {
        $res[$kr] = $vr;
        $right    = array_slice($right, 1, count($right), true);
    }
    return $res;
}
```

Questão 3 (2,5 pontos)

Desenvolva um algoritmo (sequência de passos por escrito) e o implemente como uma função PHP para encontrar a melhor aproximação inteira da raíz quadrada de um determinado número também inteiro recebido como parâmetro de entrada. Atenção: não utilize métodos prontos do PHP de cálculo de raiz quadrada.

Exemplos:

- a melhor aproximação inteira da raiz quadrada do número 10 é 3;
- a melhor aproximação inteira da raiz quadrada do número 16 é 4;
- a melhor aproximação inteira da raiz quadrada do número 7 é 3;
- a melhor aproximação inteira da raiz quadrada do número 110 é 10;

R:

- N <- <entrada do teclado>
- I <- 1
- SE N < 0
  - RETORNE ERRO
- ENQUANTO I * I < N
  - I <- I + 1
- SE |I*I - N| < |(I-1)*(I-1) - N|
  - RETORNE I
- RETORNE I-1

Em PHP (obs: uma das inúmeras possibilidades):

```
<?php
     function raiz_inteira($n) {

          $i = $n < 0 ? exit('Numero invalido') : 1;

          while ($i++ * $i < $n);

          return abs($i * $i - $n) < abs(($i-1)*($i-1) - $n) ?
                 $i : $i - 1;
     }
?>
```

Questão 4 (3,5 pontos)

Escreva uma função que recebe uma string e retorna **`TRUE`** caso hajam palavras repetidas na string ou **`FALSE`** caso não encontre palavras idênticas. Considere que a string pode conter qualquer número de palavras e que as palavras são separadas por caracteres de espaço ou de pontuação (tais como **`,`**, **`!`**, **`?`**, **`(`**, **`)`**, **`-`**, **`:`**, **`;`**, **`.`**). Portanto, sua função deve identificar as palavras contidas na string e verificar se existem duas ou mais iguais. Considere ainda que a mesma palavra pode ser escrita usando letras maiúsculas ou minúsculas e ainda assim ser considerada a mesma palavra.

R:

```
function ha_repeticao($string) {

   $arr = preg_split('/[\s,!?()-:;.)]/', strtoupper($string));

   for($i = 0; $i < count($arr); $i++)
      for($j = $i + 1; $j < count($arr); $j++)
           if($arr[$i] == $arr[$j]) return true;

   return false;
}
```