Fundação CECIERJ – Vice Presidência de Educação Superior à Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação II
AD1 –1º Semestre de 2014

Nome:_____

1.(2.5 ptos) Suponha a criação de um site para uma disciplina presencial com aulas realizadas todas as segundas e quartas-feiras no período entre 3 de fevereiro e 5 de julho de 2014.

Escreva um programa em PHP para computar e exibir os dias de aula dessa disciplina em uma página HTML completando a frase: ***"Dias de aula: seg 03/02, qua 05/02, seg 10/02, qua ..."***. Ou seja, as datas deverão ser exibidas no formato **seg ou qua *DD/MM***, separadas entre si por vírgulas. Atenção: depois do último dia de aula deve haver um ponto final e não uma vírgula.

GABARITO - QUESTÃO 1

```php
<?php
$inicio = strtotime('03-02-2014');
$fim = strtotime('05-06-2014');
$segundosDia = 24*60*60;
$strDias = "Dias de aula:";

$aux = $inicio;
while($aux <= $fim) {
    if(date('w', $aux) == 1) {
        $strDias .= ' seg '.date('d/m',$aux).',';
        $aux+=2*$segundosDia;
    } else if(date('w', $aux) == 3) {
        $strDias .= ' qua '.date('d/m',$aux).',';
        $aux+=5*$segundosDia;
    }
}

echo rtrim($strDias, ",").".";
?>
```

2. (5 ptos) Uma fábrica de sapatos deseja medir a produtividade dos seus operários.

a) (1.5 ptos) Escreva uma função PHP que recebe:
- um array descrevendo pares ***[nomeDoOperario] => numSapatosProduzidos***;
- um segundo array descrevendo ***[nomeDoOperario] => totalHorasTrabalhadas***;

Sua função deve retornar o tempo médio necessário para se produzir um sapato pelo grupo de operários.

GABARITO - QUESTÃO 2 - ITEM A

```php
<?php
//RESPOSTA
function getProdutividade($arrSapatos, $arrHoras) {
    $totalSapatos = 0;
    foreach ($arrSapatos as $operario => $sapatos) {
        $totalSapatos += $sapatos;
    }

    $totalHoras = 0;
    foreach ($arrHoras as $operario => $horas) {
        $totalHoras += $horas;
    }

    return $totalSapatos / $totalHoras;
}

//EXEMPLO DE USO
$arrSapatos = Array("joao" => 3, "jose" => 2);
$arrHoras = Array("joao" => 1, "jose" => 2);
echo getProdutividade($arrSapatos, $arrHoras);
?>
```

b)(1 pto) Escreva uma função PHP chamada **atualizaProducao** que recebe:

- um nome de um operário;
- um número contabilizando novos sapatos produzidos por esse operário;
- um array descrevendo pares ***[nomeDoOperario] => numSapatosProduzidos.***

Sua função deve atualizar o array recebido no par correspondente ao operário acrescentando o número de novos sapatos produzidos por ele no array. Caso não exista um par com o nome correspondente, este par deve ser criado no array.

GABARITO - QUESTÃO 2 - ITEM B

```php
<?php
//RESPOSTA
function atualizaProducao($operario, $sapatos, &$producao) {
    if(array_key_exists($operario, $producao)) {
        $producao[$operario] += $sapatos;
    } else {
        $producao[$operario] = $sapatos;
    }
}

//EXEMPLO
$producao = Array("joao" => 3, "jose" => 2);
atualizaProducao("joao", 7, $producao);
echo $producao["joao"];
?>
```

c)(1.5 ptos) Escreva uma função PHP chamada **ordenaPorHorasTrabalhadas** que recebe um array descrevendo pares [nomeDoOperario] => totalHorasTrabalhadas. Sua função deve ordenar o array por ordem crescente do número de horas trabalhadas pelos operários.

GABARITO - QUESTÃO 2 - ITEM C

```php
<?php
//RESPOSTA
function ordenaPorHorasTrabalhadas(&$arrHoras) {
    asort($arrHoras);
}

//EXEMPLO
$arrHoras = Array("joao" => 1, "jose" => 2, "adao" => 4);
print_r($arrHoras);
?>
```

d)(1 pto) Escreva uma função PHP que recebe:

- um array descrevendo pares [nomeDoOperario] => numSapatosProduzidos;
- um segundo array descrevendo [nomeDoOperario] => totalHorasTrabalhadas;
- uma string contendo o nome de um operário;

e retorna a produtividade desse operário, medida como o número de sapatos por hora. Caso o operário não seja encontrado, deve retornar -1.

GABARITO - QUESTÃO 2 - ITEM D

```php
<?php
//RESPOSTA
function getProdutividadeOperario($arrSapatos, $arrHoras, $operario) {
    if (!(array_key_exists($nomeOperario, $arrSapatos) &&
            (array_key_exists($nomeOperario, $arrHoras)))) {

        return -1;
    }

    return $arrSapatos[$nomeOperario] / $arrHoras[$nomeOperario];
}

//EXEMPLO
$arrSapatos = Array("joao" => 3, "jose" => 2);
$arrHoras = Array("joao" => 2, "jose" => 2);
echo getProdutividadeOperario($arrSapatos, $arrHoras, "joao");
?>
```

3. (2.5 ptos) Sobre programação orientada a objetos em PHP, o que ocorre ao se declarar um método como ***static***? E uma propriedade? Exemplifique suas utilidades com um código de exemplo, descrevendo em que contexto seria utilizado.

GABARITO - QUESTÃO 3

A palavra reservada 'static' é usada para informar ao compilador PHP que um método ou atributo de uma classe é estático. Isso significa que pertencem à classe e não a um determinado objeto.

Ao pensar nos atributos estáticos, passam a ser únicos, ou seja, não são criadas cópias do atributo uma por instância da classe como de costume. A vantagem de pertencer a classe é permitir que seja mantida informação sobre a classe como um todo, enquanto normalmente (sem o uso de *static)* os atributos modelam propriedades de um único objeto.

Uma vez declarado como 'static' o método, ou atributo, pode ser acessado diretamente da classe em questão, sem a necessidade de instanciar um objeto. Desta maneira um método estático pode ser chamado antes mesmo da existência de qualquer instância da determinada classe a que pertence.

Seguem abaixo alguns exemplos de utilização.

O primeiro exemplo ilustra um atributo no qual se deseja contar quantas instâncias uma determinada classe possui. Se declarado sem o atributo estático múltiplas cópias do contador seriam criadas, uma para cada objeto da classe, não realizando a tarefa desejada. Já como estático, um único contador é criado. Neste exemplo o mesmo contador é acessado nos construtores das instâncias para ser incrementado e nos destrutores para ser decrementado. Cabe notar o uso do operador :: para acessar o método consultaContador diretamente da

classe, e o uso da palavra reservada self para que uma instância acesse uma propriedade estática de sua classe.

```php
<?php
class UsuariosCadastrados {
    static $contadorUsuarios = 0;
    var $nome;

    function __construct($nome) {
        self::$contadorUsuarios++;
        $this->nome = $nome;
    }

    function setNome($nome) {
        $this->nome = $nome;
    }

    function getNome() {
        return $this->nome;
    }

    static function getContadorUsuarios() {
        return self::$contadorUsuarios;
    }

    function __destruct() {
        self::$contadorUsuarios--;
    }
}

$joao = new UsuariosCadastrados("joao");
$jose = new UsuariosCadastrados("jose");
echo UsuariosCadastrados::getContadorUsuarios();
?>
```

Em um segundo exemplo de utilização a classe StringUtils contém três métodos para facilitar a manipulação de Strings e um atributo que é usado por um desses métodos. Para a função que verifica se uma String se trata de um nome próprio, consideramos que nomes próprios devem conter apenas palavras com, no mínimo, 3 letras, separadas por espaços em branco.

```php
<?php
class StringUtils {
    static $regexNomeProprio = "/^([A-Za-z]{3,} ?)+$/";

    static function startsWith($str, $start) {
        return ($start === "" || strpos($str, $start) === 0);
    }

    static function endsWith($str, $end) {
        return ($end === "" || strpos($str, $end) === strlen($str) - strlen($end));
    }

    static function isNomeProprio($str) {
        return preg_match(self::$regexNomeProprio, $str);
    }
}

$nome = "Joao Marques Silva";
if(StringUtils::isNomeProprio($nome)) {
    echo $nome." &eacute; nome pr&oacute;prio.<br />";
} else {
    echo $nome." n&atilde;o &eacute; nome pr&oacute;prio.<br />";
}

$addr = "http://www.cederj.edu.br/";
if(StringUtils::startsWith($addr, "http://") &&
                                        StringUtils::endsWith($addr, ".br/")) {

    echo $addr." &eacute; uma URI cujo dom&iacute;nio &eacute; brasileiro.<br />";
} else {
    echo $addr." n&atilde;o &eacute; uma URI cujo dom&iacute;nio &eacute; "
                                                        ."brasileiro.<br />";
}
?>
```