1)	**3,0**	
2)	**3,5**	
3)	**3,5**	

Fundação CECIERJ – Vice Presidência de Educação Superior à Distância Curso de Tecnologia em Sistemas de Computação Disciplina: Programação de Aplicações Web
Professores: Diego Passos e Uéverton dos Santos Souza
Gabarito da AD1 – 1º Semestre de 2019

Esta AD avalia o uso das estruturas básicas de repetição e condição, a criação e uso de funções, manipulação de vetores e programação orientada a objetos na linguagem PHP.

Suponha que a empresa em que você trabalha foi contratada para desenvolver um sistema de controle de estoque e vendas para um supermercado em PHP. As questões desta AD solicitam que você desenvolva ou analise partes do código desse sistema.

1) **(3,0 pontos)** Suponha que você tenha sido encarregado de escrever uma função que calcule, para cada cliente, quanto foi seu gasto total no mês. A função recebe como parâmetro a lista das compras realizadas no mês por todos os clientes do supermercado. Essa lista é representada por dois parâmetros: um vetor com os identificadores dos clientes (cada cliente tem um identificador numérico único) que realizaram cada compra e outro com os valores de cada compra. Assim, para obter os dados da *i-ésima* compra, a função deve consultar a *i-ésima* posição do primeiro vetor para descobrir o cliente que fez a compra e a *i-ésima* posição do segundo vetor para descobrir o valor da compra.

Nestas condições, a função deve retornar um novo vetor indexado pelo identificador dos clientes em que cada posição contenha a soma de todas as compras realizadas por aquele cliente no mês. Escreva essa função em PHP.

R:

```
function totais_por_cliente($identificadores, $valores) {
     $retorno = array();
     for($i = 0; $i < count($identificadores); $i++) {
          if(!array_key_exists($identificadores[$i], $retorno))
               $retorno[$identificadores[$i]] = $valores[$i];
          else
               $retorno[$identificadores[$i]] += $valores[$i];
     }
     return $retorno;
}

$identificadores = array(12345612312, 21512352312, 12345612312);
$valores = array(80.55, 20.40, 15.50);
print_r(totais_por_cliente($identificadores, $valores));
```

Observação: não será cobrada a verificação com array_key_exists na resposta, apesar do aviso do tipo NOTICE.

2) **(3,5 pontos)** Suponha agora que você seja encarregado de desenvolver algumas funções relacionadas à parte de cadastro de novos clientes. Em particular, você deve desenvolver uma função que teste se o CPF provido pelo usuário é válido. Você recebe instruções para que essa verificação seja feita em três partes:

1. Uma função que verifique se o formato (isto é, quantidade de algarismos e caracteres separadores) do CPF está correta.

2. Uma função que remova qualquer separador de um CPF com formato válido, se esse existir (isto é, a função deverá remover quaisquer caracteres do CPF que não sejam algarismos decimais).

3. Uma função que faça a validação dos dígitos verificadores do CPF.

Nestas condições, pede-se:

a) **(1,0 ponto)** Escreva em PHP a função de verificação de formato **usando expressões regulares**. Um CPF tem formato válido se for da forma DDD.DDD.DDD-DD ou DDDDDDDDDDD, onde cada ‘D’ denota um algarismo decimal. Sua função deverá receber o CPF na forma de uma string e retornar 1, caso o formato esteja correto ou 0 caso contrário.

R:

```
function validaCPF($cpf) {
     return preg_match('/\d{11}|\d{3}\.\d{3}\.\d{3}-\d{2}/',
                       $cpf) ? 1 : 0;
}

echo validaCPF('123.456.789-00');
echo validaCPF('123.456.789-0');
echo validaCPF('123.456789-00');
echo validaCPF('12345678900');
```

b) **(1,0 ponto)** Escreva em PHP uma função que receba um CPF na forma de uma string e remova os pontos e traço (se houver), retornando uma nova string composta apenas pelos números.

R:

```
function limpaCPF($cpf) {
     return str_replace('-', '', str_replace('.', '', $cpf));
}
```

c) **(1,5 pontos)** Escreva em PHP uma função que verifique se os dígitos verificadores de um CPF (recebido na forma de uma string contendo apenas os números) estão corretos. Os dígitos verificadores de um CPF são os dois últimos e são calculados da seguinte forma:

1. Pegue os 9 primeiros dígitos do CPF e multiplique cada um por pesos decrescentes, começando do 10 (ou seja, o primeiro dígito do CPF será multiplicado por 10, o segundo por 9 e assim sucessivamente).

2. Some os resultados das multiplicações do passo anterior.

3. Faça a divisão inteira dessa soma por 11, e guarde o resto.

4. Se o resto da divisão for:
    a. Menor que 2, então o primeiro dígito verificador é 0.
    b. Maior ou igual 2, então o primeiro dígito verificador é **(11 - resto)**.

Segundo dígito verificador:

5. Pegue os 10 primeiros dígitos do CPF (incluindo o primeiro dígito verificador) e multiplique cada um por pesos decrescentes, começando do 11 (ou seja, o

primeiro dígito do CPF será multiplicado por 11, o segundo por 10 e assim sucessivamente).

6. Some os resultados das multiplicações do passo anterior.

7. Faça a divisão inteira dessa soma por 11, e guarde o resto.

8. Se o resto da divisão for:
    a. Menor que 2, então o segundo dígito verificador é 0.
    b. Maior ou igual 2, então o segundo dígito verificador é **(11 - resto)**.

R:

```
<?php
function checarCPF($cpf) {
     $cpf = limpaCPF($cpf);
     $checksum1 = 0; $checksum2 = 0;
     for($i = 0; $i < 9; $i++) {
          $checksum1 += (10 - $i) * $cpf[$i];
          $checksum2 += (11 - $i) * $cpf[$i];
     }
     $checksum2 += 2*$cpf[9];

     $resto1 = $checksum1 % 11;
     $digito1 = $resto1 < 2 ? 0 : 11 - $resto1;
     $resto2 = $checksum2 % 11;
     $digito2 = $resto2 < 2 ? 0 : 11 - $resto2;

     return $cpf[9] == $digito1 && $cpf[10] == $digito2;
}

var_dump(checarCPF('111.222.333-44'));
var_dump(checarCPF('111.222.333-96'));
var_dump(checarCPF('11436125740'));
```

3) **(3,5 pontos)** Considere agora a parte do sistema que diz respeito ao controle de estoque. Suponha que você seja encarregado de criar uma classe em PHP que modele o estoque. Sua classe deverá armazenar um conjunto de produtos presentes no estoque e sua quantidade. Sua classe deverá ainda fornecer métodos para a adição de novos produtos ao estoque e para a remoção de produtos do estoque (por exemplo, para o caso de uma venda).

Além disso, cada produto por sua vez deve ser representado por uma segunda classe `Produto`. São informações necessárias uma descrição textual, um tipo (considere os

tipos: bebida, alimento perecível, alimento não perecível e produtos de limpeza), um prazo de validade (em número de dias) e um valor unitário. A classe deverá prover métodos para a mudança do valor unitário.

Implemente ambas as classes em PHP.

Observação: você não precisa implementar outros métodos ou atributos, além dos que foram listados no enunciado.

R:

```
class Estoque {

     private $produtos;

     public function __construct() {
          $this->produtos = array();
     }

     public function adicionarProduto($produto) {
          if(!array_key_exists($produto->descricao(),
                               $this->produtos))
             $this->produtos[$produto->descricao()] = array();

          $this->produtos[$produto->descricao()] []= $produto;
     }

     public function removerProduto($descricao) {
          return array_shift($this->produtos[$descricao]);
     }

     public function quantidade($descricao) {
          if(!array_key_exists($descricao, $this->produtos))
               return 0;
          return count($this->produtos[$descricao]);
     }
}

class Produto {

     private $descricao_textual;
```

```
    private $tipo;
    private $prazo_validade;
    private $valor_unitario;

    public function __construct($descricao_textual, $tipo,
$prazo_validade, $valor_unitario) {
        $this->descricao_textual = $descricao_textual;
        $this->tipo = $tipo;
        $this->prazo_validade = $prazo_validade;
        $this->valor_unitario = $valor_unitario;
    }

    public function mudarValorUnitario($novo_valor) {
        $this->valor_unitario = $novo_valor;
    }

    public function descricao() {
        return $this->descricao_textual;
    }
}

$laranja = new Produto('laranja', 'alimento perecível', 7,
0.30);
$laranja2 = new Produto('laranja', 'alimento perecível', 7,
0.30);
$esponja_aco = new Produto('esponja de aço', 'produtos de
limpeza', 180, 5.50);

$estoque = new Estoque();
$estoque->adicionarProduto($laranja);
$estoque->adicionarProduto($laranja2);
$estoque->adicionarProduto($esponja_aco);

echo $estoque->quantidade('laranja');
echo $estoque->quantidade('maçã');
echo $estoque->quantidade('esponja de aço');

print_r($estoque->removerProduto('laranja'));

echo $estoque->quantidade('laranja');
```