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: Flavio Seixas e Marcos Lage
AD1 – 2º Semestre de 2019 - Gabarito

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 bancário 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 o saldo da conta do cliente. A função recebe como parâmetro a lista de saques e depósitos. Essa lista é representada por dois parâmetros: um vetor com as identificações dos clientes e o valor da operação. Cada cliente tem um identificador numérico único. Para o valor da operação, valores negativos representam saques, e valores positivos depósitos. Assim, para obter a *i-ésima* operação, a função deve consultar a *i-ésima* posição do primeiro vetor para descobrir o identificador do cliente e a *i-ésima* posição do segundo vetor para descobrir o valor da operação.

Nestas condições, a função deve retornar um novo vetor indexado pelo identificador dos clientes em que cada posição e o saldo da conta do cliente. Escreva essa função em PHP.

```
function calcularSaldo(array $cliente, array $operacao) {

    $cliente_unique = array_unique($cliente);

    foreach($cliente_unique as $c) {
        $saldo = 0;
```

```
        for($i = 0; $i < sizeof($cliente); $i++) {

            if ($c == $cliente[$i]) {
                $saldo += $operacao[$i];
            }
        }

        $saldo_unique[$c] = $saldo;
    }

    return $saldo_unique;
}

$cliente = array(1, 2, 1, 3, 1);
$operacao = array(200, 300, 100, 500, 400);

print_r(calcularSaldo($cliente, $operacao));
```

2) **(3,5 pontos)** A próxima tarefa é desenvolver uma função em PHP para avaliar se a senha definida pelo cliente segue a política de segurança do banco, isto é, você deve desenvolver uma função que teste se senha definida pelo cliente é forte. A verificação se uma senha é forte pode ser feita em quatro etapas desenvolvidas separadamente:

1. Uma função que verifique se a quantidade de caracteres da senha é igual ou superior a 6 caracteres.
2. Uma função que verifique se há pelo menos um caractere em letra maiúscula.
3. Uma função que verifique se há pelo menos um caractere numérico.
4. Uma função que verifique se há pelo menos um caractere especial. No caso, os caracteres especiais estão limitados ao conjunto listado abaixo.

! @ # $ % & * ( ) [ ] { }

A cada etapa de verificação de senha, a respectiva função deverá retornar um booleano: **True** se o critério for verificado, **False** caso contrário. A função de teste de senha deverá receber o resultado da verificação de cada etapa, e retornar **True** se a senha é forte, **False** caso contrário. No caso da senha não passar nos critérios de verificação, a função deverá exibir qual ou quais testes retornaram **False**.

```
function verificaEtapa1($senha) {
    if (strlen($senha) >= 6) {
```

```
        return true;
    }
    return false;
}

function verificaEtapa2($senha) {
    if(preg_match('/[A-Z]/', $senha)) {
        return true;
    }
    return false;
}

function verificaEtapa3($senha) {
    if(preg_match('/\d/', $senha)) {
        return true;
    }
    return false;
}

function verificaEtapa4($senha) {
    if(preg_match('/(\!|\@|\#|\$|\%|\&|\*|\(|\)|\[|\]|\{|\})/', $senha)) {
        return true;
    }
    return false;
}

function verificaSenha($senha) {
    $e = true;

    if (verificaEtapa1($senha) == false) {
        echo "Erro: quantidade de caracteres da senha deve ser maior que 6.\n";
        $e = false;
    }

    if (verificaEtapa2($senha) == false) {
        echo "Erro: a senha deve possuir pelo menos um caractere em maiúsculo.\n";
        $e = false;
```

```
    }

    if (verificaEtapa3($senha) == false) {
        echo "Erro: a senha deve possuir pelo menos um caractere numérico.\n";
        $e = false;
    }

    if (verificaEtapa4($senha) == false) {
        echo "Erro: a senha deve possuir pelo menos um dos seguintes caracteres especiais ! @ # $ % & ( ) [ ] { }.\n";
        $e = false;
    }

    return $e;
}

$senha = "Flavio";
verificaSenha($senha);
```

3) **(3,5 pontos)** Considere a parte do sistema que diz respeito ao registro da conta do cliente no banco. Suponha que você seja encarregado de criar uma classe em PHP que modele a Conta do cliente. Sua classe deverá armazenar o Cliente, o conjunto de operações de Saque e Depósito, e a Data de cada operação. Sua classe deverá ainda fornecer métodos para o registro de novos saques, novos depósitos, e a exibição do saldo final da conta.

Além disso, cada Cliente deverá ser representado por uma segunda classe em PHP Cliente. Esta classe deverá armazenar o nome do cliente, o seu CPF, e a categoria do cliente. As seguintes categorias são permitidas: Rubi, Ouro, Prata, Standard e Universitário. Esta classe deverá fornecer métodos para a exibição do nome, CPF e categoria do cliente, bem como um método para mudança de categoria do cliente.

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

```
abstract class Categoria {
    const Rubi = 'Rubi';
    const Ouro = 'Ouro';
    const Prata = 'Prata';
    const Standard = 'Standard';
    const Universitario = 'Universitario';
```

```
}

class Cliente {

    private $nome;
    private $CPF;
    private $categoria;

    function __construct($nome, $CPF, $categoria) {
        $this->nome = $nome;
        $this->CPF = $CPF;
        $this->categoria = $categoria;
    }

    function exibirNome() {
        echo $this->nome."\n";
    }

    function exibirCPF() {
        echo $this->CPF."\n";
    }

    function exibirCategoria() {
        echo $this->categoria."\n";
    }
}

class Operacao {
    private $data;
    private $valor;

    function __construct($data, $valor) {
        $this->data = $data;
        $this->valor = $valor;
    }

    function obterValor() {
```

```
        return $this->valor;
    }
}

class Conta {

    private $cliente;
    private $operacao = array();

    function __construct($cliente) {
        $this->cliente = $cliente;
    }

    function registraOperacao($operacao) {
        $this->operacao[] = $operacao;
    }

    function saldoFinal() {
        $saldo = 0;

        foreach($this->operacao as $op) {

            $saldo += $op->obterValor();
        }

        return $saldo;
    }
}

$cliente = new Cliente("Flavio", "12345", Categoria::Rubi);

$op1 = new Operacao("01/08/2019", 100);
$op2 = new Operacao("02/08/2019", -50);
$op3 = new Operacao("03/08/2019", 20);

$conta = new Conta($cliente);
$conta->registraOperacao($op1);
```

```
$conta->registraOperacao($op2);
$conta->registraOperacao($op3);

$cliente->exibirNome();
$cliente->exibirCPF();
$cliente->exibirCategoria();
echo "O saldo final é ".$conta->saldoFinal()."\n";
```