Q1	2,0	
Q2	2,0	
Q3	2,0	
Q4	2,0	
Q5	2,0	

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
AP1 – 2º Semestre de 2018 - Gabarito
Nome:_____

Questão 1: O ISBN é um sistema de numeração internacional que serve para identificar livros. Embora haja outros detalhes, de maneira simplificada o ISBN encontrado em um livro é uma *string* composta de 6 elementos **separados por hífens**:

1. a string **"**`ISBN`**"** (sem as aspas);
2. o elemento de prefixo, que é um número sempre com **três dígitos decimais**;
3. o elemento do grupo de registro, que é um **número decimal com pelo menos um e no máximo cinco dígitos**;
4. o elemento registrante, que é um **número decimal com pelo menos um e no máximo sete dígitos**;
5. o elemento de edição, que é um **número decimal com pelo menos um e no máximo seis dígitos**; e
6. o dígito de controle, que é **um único dígito decimal**.

Um exemplo de string representando um ISBN é: **"**`ISBN-978-0-571-08989-5`**"** (novamente, sem as aspas).

Escreva uma expressão regular em PHP que identifique se uma dada string está no formato correto de um ISBN, **considerando a descrição acima**.

Uma possível resposta é a expressão regular abaixo:

```
^ISBN-[0-9]{3}-[0-9]{1,5}-[0-9]{1,7}-[0-9]{1,6}-[0-9]$
```

Os elementos que compõem a expressão regular acima são:

- “^ISBN”’: demarcar que a string avaliada deve necessariamente começar por “ISBN…”. Isso impede que haja outros caracteres antes do ISBN.
- “-[0-9]{3}”: define que o prefixo “ISBN” deve ser seguido de um hífen e exatamente 3 dígitos decimais.
- “-[0-9]{1,5}”: define que a próxima parte da string deve ser um hífen seguido de 1 a 5 dígitos decimais.
- “-[0-9]{1,7}”: define que a próxima parte da string deve ser um hífen seguido de 1 a 7 dígitos decimais.
- “-[0-9]{1,6}”: define que a próxima parte da string deve ser um hífen seguido de 1 a 6 dígitos decimais.
- “[0-9]$”: define que a próxima parte da string deve ser um único dígito decimal, e que esse necessariamente encerra a string.

Questão 2: Além do formato descrito na questão anterior, a validade de um ISBN depende também do valor do dígito de controle estar correto. O valor desse dígito é calculado com base nos **doze primeiros dígitos** do ISBN de acordo com o seguinte algoritmo:

1. Somam-se os valores do 1º, 3º, 5º, 7º, 9º e 11º dígitos do ISBN.
2. Somam-se os valores do 2º, 4º, 6º, 8º, 10º e 12º dígitos do ISBN e o resultado é multiplicado por 3.
3. Os resultados dos passos 1 e 2 são somados.
4. Pega-se o resultado do passo 3 e calcula-se o resto da sua divisão por 10.
5. O resultado do passo 4 é subtraído de 10.
6. Se o resultado do passo 5 for diferente de 10, esse resultado é o dígito de verificação. Caso contrário, o dígito de verificação é 0.

Por exemplo, para o ISBN “`ISBN-978-0-571-08989-5`”, o dígito de verificação é `5` porque:

```
(Passo 1) 9 + 8 + 5 + 1 + 8 + 8 = 39
(Passo 2) 3 x (7 + 0 + 7 + 0 + 9 + 9) = 96
(Passo 3) 39 + 96 = 135
(Passo 4) Resto de 135 dividido por 10 = 5
(Passo 5) 10 - 5 = 5
(Passo 6) Como 5 é diferente de 10, dígito de verificação é 5.
```

Escreva uma função em PHP que receba um ISBN na sua forma de *string* (isto é, no mesmo formato descrito na **Questão 1**) e teste se o dígito de verificação está correto. Sua função deve imprimir uma mensagem indicando o resultado do teste.

Observação: sua função **não** precisa realizar qualquer tipo de verificação em relação ao formato da *string* recebida como parâmetro. Você pode assumir que a *string* sempre estará no formato especificado na **Questão 1**.

Uma possível solução é dada a seguir:

```
function testaDigitoVerificacao($isbn) {

	$somaDigitosPares = 0;
	$somaDigitosImpares = 0;

	$l = strlen($isbn);
	$digitosLidos = 0;
	for ($i = 5; $i < $l && $digitosLidos < 12; $i++) {

		if ($isbn[$i] == "-") continue ;

		$digitosLidos = $digitosLidos + 1;
		if ($digitosLidos % 2 == 0) $somaDigitosPares = $somaDigitosPares + $isbn[$i];
		else $somaDigitosImpares = $somaDigitosImpares + $isbn[$i];
	}

	$somaDigitosPares = 3 * $somaDigitosPares;
	$somaTotal = $somaDigitosPares + $somaDigitosImpares;
	$resultadoModulo = $somaTotal % 10;
	$resultadoSubtracao = 10 - $resultadoModulo;

	if ($resultadoSubtracao == 10) $resultadoSubtracao = 0;

	if ($resultadoSubtracao == $isbn[$l - 1]) echo "Digito eh valido!\n";
	else echo "Digito eh invalido!\n";

}
```

A solução acima é simplesmente uma implementação em php do pseudo-código dado no enunciado. Além das simples operações aritméticas do algoritmo, deve-se reparar no *loop* do tipo *for* utilizado. Ele basicamente itera pelos caracteres da *string* recebida como parâmetro (o ISBN). O índice da repetição começa em 5, porque os 5 primeiros caracteres da *string* (índices 0 a 4) sempre serão “ISBN-”. A cada iteração, testa-se

também se o caractere atual é hífen: nesse caso, o caractere é ignorado, passando-se para a próxima iteração do *loop*. Caso contrário, trata-se de um dígito. Cada dígito encontrado é contado pela variável `$digitosLidos` e seu valor é usado para discernir dígitos pares e ímpares (para efeito das somas dos passos 1 e 2 do pseudo-código). Quando 12 dígitos já tiverem sido lidos, a repetição é encerrada e as demais operações aritméticas são realizadas.

Questão 3: Considere o seguinte código em PHP:
<?php

```php
<?php
function teste($a, $b) {

	$contador = array();
	for ($i = 0; $i < count($a); $i = $i + 1) {

		if (isset($contador[$a[$i]]))
			$contador[$a[$i]] = $contador[$a[$i]] + 1;
		else
			$contador[$a[$i]] = 1;
		$elementos[$b[$i]] = 1;
	}
	$maximo = 0;
	$maximal = "";
	foreach ($contador as $i => $v) {
		if ($maximo < $v) {
			if (isset($elementos[$i])) {
				$maximo = $v;
				$maximal = $i;
			}
		}
	}
	echo "$maximal -> $maximo";
}

$v1[] = "Um";
$v1[] = "Dois";
$v1[] = "Tres";
$v1[] = "Dois";
$v1[] = "Quatro";
$v1[] = "Quatro";
$v1[] = "Cinco";
$v1[] = "Dois";

$v2[] = "Quatro";
```

```php
$v2[] = "Um";
$v2[] = "Cinco";
$v2[] = "Seis";
$v2[] = "Sete";
$v2[] = "Oito";
$v2[] = "Nove";
$v2[] = "Dez";

teste($v1, $v2);
?>
```

Sabendo que a função `isset()` verifica se um certo índice existe em um *array*, responda:

a) **(1,0 ponto)** Quando executado, o que esse código imprime?

Realizando a execução do código da função `teste()` passo a passo, considerando os conteúdos dos vetores `$v2` e `$v1` passados como parâmetro, o código imprime:

```
Quatro -> 2
```

b) **(1,0 ponto)** Repare que a função teste() é chamada passando-se os *arrays* `$v1` e `$v2` como parâmetros. Repare ainda que ambos os vetores são inicializados com a mesma quantidade de elementos. Caso façamos uma pequena alteração no código, inicializando o vetor $v2 com um elemento a menos, **o código resultante ainda é executado, porém o PHP acusa um *warning***. Explique por que esse *warning* acontece.

Note que a primeira repetição usada na função teste() utiliza o índice $i variando de 0 ao número de elementos do parâmetro $a (correspondente ao vetor $v1, na chamada em questão). No entanto, dentro dessa repetição, esse mesmo índice $i é usado para acessar os elementos do parâmetro $b (correspondente ao vetor $v2, na chamada em questão). No código original, como $v1 e $v2 (e, consequentemente, $a e $b) têm o mesmo número de elementos, não há problema. No entanto, na hipótese sugerida no enunciado, $b teria um elemento a menos que $a (7 contra 8). Logo, na iteração em que $i é 7, o código realiza uma tentativa de leitura a uma posição não inicializada do vetor $b, resultando em um *warning*.

Questão 4: Escreva uma função em PHP que receba como parâmetros duas matrizes `A` e `B` de mesmas dimensões e calcule uma terceira matriz `C` em que cada elemento `(i,j)` é igual à soma dos elementos `(i,j)` de `A` e de `B`.

Observação: sua função pode assumir que as matrizes A e B possuem as mesmas dimensões. Não é necessário validar a entrada.

Uma possível solução é dada pelo código a seguir:

```
function somaMatrizes($a, $b) {

    $linhas = count($a);
    $colunas = count($a[0]);

    $c = array();

    for ($i = 0; $i < $linhas; $i++) {

        $c[$i] = array();

        for ($j = 0; $j < $colunas; $j++) {

              $c[$i][$j] = $a[$i][$j] + $b[$i][$j];
        }
    }

    for ($i = 0; $i < $linhas; $i++) {
        for ($j = 0; $j < $colunas; $j++) {

              echo $c[$i][$j] . " ";
        }
        echo "<br>";
    }
}
```

Repare que o enunciado não pede que a matriz $c seja impressa, apenas calculada. A impressão (segundo bloco de repetições aninhadas) foi incluída nesse trecho apenas de maneira ilustrativa.

Questão 5: Conforme visto na disciplina, a linguagem PHP permite programação orientada a objetos. Suponha que você esteja desenvolvendo uma aplicação armazenamento de arquivos na nuvem (algo similar ao Dropbox). Em particular, como parte da sua aplicação, você precisa escrever uma classe que represente um arquivo armazenado. Essa classe deverá incluir:

- Uma *string* representando o nome do arquivo.
- Uma *string* representando o diretório no qual o arquivo está armazenado no servidor.

- Um número inteiro representando o tamanho do arquivo.
- Uma data representando a última modificação do arquivo.
- Um *array* de strings, representando os nomes dos usuários que têm permissão para acessar aquele arquivo.
- Um método que recebe como parâmetro um nome de usuário, verifica se o mesmo tem permissão para acessar o arquivo e, em caso positivo, atualiza a data de acesso ao arquivo.
- Um método que atualiza o tamanho do arquivo.

Outras informações/métodos não são necessários.

Uma possível declaração da classe atendendo aos requisitos estabelecidos no enunciado é:

```
class Arquivo {

    private $nome = "";
    private $diretorio = "";
    private $tamanho = 0;
    private $ultimaModificacao;
    private $usuariosComPermissao = array();

    function atualizaArquivo($usuario) {

        for ($i = 0; $i < count($this->usuariosComPermissao); $i++) {

              if ($usuario == $this->usuariosComPermissao[$i]) {

                    break ;
              }
        }

        if ($i < count($this->usuariosComPermissao)) {

              $this->ultimaModificacao = getDate();
        }
    }

    function atualizaTamanho($novoTamanho) {

        $this->tamanho = $novoTamanho;
    }
}
```