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

Curso de Tecnologia em Sistemas de Computação

Disciplina: Programação II

AP1 2015/1 - GABARITO

Nome – _____

Assinatura – _____

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

Questão 1 (3.0 pontos)

A orientação a objetos é um modelo de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema de software. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos.

(Fonte: Wikipédia)

Com base no paradigma de programação conhecido como Orientação a Objetos, responda, com suas palavras, os itens abaixo:

a. (1.5 pontos) O que significa em PHP adicionar as palavras reservadas *public*, *private* ou *protected* à declaração de um método, dentro de uma classe? Dê exemplos de uso de cada uma em sua argumentação.

   RESPOSTA:

Essas palavras reservadas são usadas na orientação a objetos para alterar a visibilidade de um método, de acordo com as descrições abaixo:
- *public*: significa que o método é visível (poderá ser chamado) a partir de qualquer classe do PHP;
- *private*: significa que o método é visível (poderá ser chamado) apenas a partir da própria classe que o define;
- *protected*: significa que o método só é visível (poderá ser chamado) de dentro da própria classe que o define e por qualquer classe que estende a classe que o define. Ela é usada no contexto de herança.

b. (1.5 pontos) O que significa em PHP adicionar a palavra *static* à declaração de um atributo, dentro de uma classe? Dê exemplos de uso em sua argumentação.

RESPOSTA:
A palavra reservada *static* antes de um atributo é usada para indicar que esse atributo pertence à classe, e não a instância da mesma. Em outras palavras, significa dizer que não é preciso criar uma instância da classe para ler ou alterar o conteúdo dessa variável. Portanto há apenas uma cópia de atributos marcados com *static,* compartilhada pela classe.

Questão 2 (3.0 pontos)

Suponha que um usuário de um sistema genérico digite sua data de nascimento em um campo *input* do tipo *text* no formato de datas mais comumente usado no brasil. Ex: **20/05/1987**. Escreva em PHP uma função que receba como parâmetro a data digitada pelo usuário e retorne sua idade.

RESPOSTA:

```
function calculaIdade($dtNascimento) {
	$partesNascimento = explode("/", $dtNascimento);
	$partesHoje = explode("/", date("d/m/Y"));

	$diferencaAnos = $partesHoje[2] - $partesNascimento[2];

	$diferencaMeses = $partesHoje[1] - $partesNascimento[1];

	if ($diferencaMeses > 0) {
		return $diferencaAnos;
	} else if ($diferencaMeses < 0) {
		return $diferencaAnos-1;
	} else {
		$diferencaDias = $partesHoje[0] - $partesNascimento[0];
```

```
            if ($diferencaDias >= 0) {
                return $diferencaAnos;
            } else {
                return $diferencaAnos-1;
            }
        }

        return $idade;
}
```

Questão 3 (4.0 pontos)

Em um tradicional campeonato de futebol brasileiro, cada uma de suas fases é marcada pelo enfrentamento de pares de equipes participantes duas vezes, nos chamados jogos de ida e jogos de volta. Suponha que você esteja implementando um sistema para divulgação dos dados das partidas de futebol desse campeonato e que, dentro do código do seu sistema, exista a variável **$partidas** preenchida, inicialmente, com o seguinte código:

```php
<?
class Partida {
    public $time1;
    public $time2;
    public $dia;
    public $placarTime1;
    public $placarTime2;

    public function __construct($time1, $time2, $dia) {
        $this->time1 = $time1;
        $this->time2 = $time2;
        $this->dia = $dia;
    }
}

$partidas = Array();
array_push($partidas, new Partida("Vasco", "Flamengo", "2015-03-07"));
array_push($partidas, new Partida("Botafogo", "Fluminense", "2015-03-14"));
array_push($partidas, new Partida("Flamengo", "Vasco", "2015-03-21"));
array_push($partidas, new Partida("Fluminense", "Botafogo", "2015-03-28"));
?>
```

Sabendo que, nesse campeonato, cada par de equipes se enfrentam somente nos jogos de ida e volta, e que novos jogos serão adicionados ao array **$partidas** sempre seguindo a ordem das datas, responda os itens a seguir:

Item A (2.0 pontos)

Escreva uma função em PHP em que, passado o array **$partidas** como entrada, ordenado pela respectiva data das partidas, retorne um novo array agrupando cada "jogo de ida" com seu respectivo "jogo de volta".

Assim, o novo array produzido apresenta a sucessão de jogos intercalando o primeiro “jogo de ida” a acontecer seguido por seu respectivo “jogo de volta”, o próximo “jogo de ida” e seu respectivo “jogo de volta”, o terceiro “jogo de ida” e seu respectivo “jogo de volta” e assim sucessivamente.

RESPOSTA:

```
function obtemPartida($partidas, $time1, $time2) {
	foreach($partidas as $partida) {
		if ($partida->time1 == $time1 && $partida->time2 == $time2) {
			return $partida;
		}
	}

	return null;
}

function montaPartidasIdaVolta($partidas) {
	$partidasIdaVolta = Array();

	foreach($partidas as $partida) {
		$partidaIda = obtemPartida($partidasIdaVolta, $partida->time1, $partida->time2);

		if ($partidaIda == null) {
			$partidaIda = obtemPartida($partidas, $partida->time1, $partida->time2);
			$partidaVolta = obtemPartida($partidas, $partida->time2, $partida->time1);

			array_push($partidasIdaVolta, $partidaIda);
			array_push($partidasIdaVolta, $partidaVolta);
		}
	}

	return $partidasIdaVolta;
}
```

Item B (2.0 pontos)

Uma outra regra importante do campeonato descrito na questão, estabelece que um jogo de volta **não deverá acontecer** caso a diferença de gols no placar no **jogo de ida** seja maior do que 2.
Sabendo dessa regra, implemente uma função em PHP que recebe como parâmetro o array no qual os jogos de ida e volta são organizados de maneira intercalada (retornado pela função implementada no **Item A)** e retorne um array contendo todos os **jogos de volta** que foram cancelados.
Assuma que jogos que ainda não aconteceram tem as variáveis $placarTime1 e $placarTime2 com o valor -1.
Observação: você não precisa ter respondido o Item A para fazer esta questão.

RESPOSTA:

```
function montaPartidasCanceladas($partidas) {
	$canceladas = Array();

	foreach($partidas as $partida) {
		$partidaIda = obtemPartida($canceladas, $partida->time1, $partida->time2);

		if ($partidaIda == null) {
			if (($partida->placarTime2 - $partida->placarTime1 > 2)
				|| ($partida->placarTime1 - $partida->placarTime2 > 2)) {

				$partidaVolta = obtemPartida($partidas, $partida->time2,
$partida->time1);

				array_push($canceladas, $partidaVolta);
			}
		}
	}

	return $canceladas;
}
```