Q1		**1,5**
Q2		**1,5**
Q3		**2,0**
Q4		**2,5**
Q5		**2,5**

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
AP1 – 1º Semestre de 2018 - Gabarito
Nome:_____

Questão 1: Suponha que você esteja escrevendo um programa em PHP para realizar análises em códigos-fonte HTML. Como parte dessa análise, você deseja identificar *tags* HTML no código-fonte. Para isso, você decide usar expressões regulares. Escreva uma expressão regular que case com *tags* HTML (**e nada mais**). Assuma que uma *tag* HTML tem sempre o formato "<nome_da_tag>" (sem as aspas), onde "nome_da_tag" pode ser qualquer sequência de letras (maiúsculas ou minúsculas) e números **com no máximo 15 caracteres**.

Solução

Uma possível expressão resposta correta seria a expressão regular /<[a-zA-Z0-9]{1,15}>/

Explicação:

- O primeiro caractere ('<') determina que a *tag* HTML deve ser iniciada por '<'.
- Em seguida, é listado um conjunto de caracteres possíveis para o nome da *tag*. Como especificado no enunciado, o nome da *tag* pode ser formado por qualquer sequência de letras minúsculas ('a-z'), maiúsculas ('A-Z') e números ('0-9').
- O conjunto supracitado é seguido pelo modificador de repetição '{1,15}', denotando que o nome da *tag* deve ter algo entre 1 e 15 caracteres (inclusive).

- O último item é o caracter '>', indicando o final da *tag*.

Note que variações são possíveis.

Questão 2: Considere os seguintes três trechos de código em PHP:

<?php
$nome = "Neymar";
function teste() {
echo "Nome: $nome";
}
teste();
?>
	<?php
	$nome = "Messi";
	function teste() {
	$nome = "Neymar";
	}
	teste();
	echo "Nome: $nome";
	?>
		<?php
		$nome = "Messi";
		function teste($nome) {
		$nome = "Neymar";
		}
		teste($nome);
		echo "Nome: $nome";
		?>

(a)	**(b)**	**(c)**

Para cada trecho, diga o que será impresso **e explique** o por quê.

Solução

A questão aborda os temas de escopo de variáveis e tipo de passagem de parâmetros na linguagem PHP. Em particular, os trechos de código imprimem (sem as aspas):

a) "Nome: "
b) "Nome: Messi"
c) "Nome: Messi"

Explicação:
No item (a), a função **teste()** tenta acessar uma variável ou parâmetro chamada(o) "nome". Embora tal variável exista e tenha um valor definido (segunda linha do código), essa definição ocorre fora da função **teste()**. Logo, ao tentar acessar a variável dentro da função **teste()**, o PHP busca uma variável local à função, algo que não foi definido anteriormente. Como resultado, é impresso um nome vazio. Repare ainda que a execução do código resulta em um aviso por parte do PHP sobre a não definição da variável local "nome".

No item (b), a impressão do valor da variável "nome" ocorre fora da função **teste()**. Repare que essa função, declarada e chamada antes da impressão, atribui o vlaor "Neymar" à uma variável também chamada "nome". No entanto, assim como no item anterior, a variável manipulada dentro da função **teste()** tem escopo local, não afetando

o valor da variável “nome” externa. Logo, mesmo após a execução da função **teste()**, o valor da variável no programa principal se mantém “Messi”.

O item (c) é bastante parecido com o item (b). No entanto, nesse caso, a função **teste()** recebe um parâmetro chamado “nome”. O programa principal, ao chamar **teste()** passa como parâmetro a sua variável chamada “nome” (nesse ponto, com o valor “Messi”). Dentro da função **teste()**, o parâmetro “nome” tem o valor “Neymar” atribuído. Deve-se notar, entretanto, que por padrão PHP utiliza passagem de parâmetro por valor. Isso significa que o parâmetro “nome” modificado dentro da função não corresponde à variável externa “nome”. Assim, essa variável não tem seu valor alterado pela chamada da função, permanecendo com o valor original “Messi” no momento em que ocorre a impressão.

Questão 3: Servidores web tipicamente armazenam *logs* das páginas acessadas. Esses *logs* associam, entre outras informações, o endereço de origem da requisição web e o endereço da página requisitada. Suponha que você esteja desenvolvendo um *software* de auditoria de servidores web. Esse *software* tem por objetivo analisar o perfil de acesso às páginas do servidor e tentar identificar usuários maliciosos (ou, ao menos, com comportamento suspeito).
Como parte desse software, você deseja desenvolver uma função que receba um vetor com os endereços de origem de cada requisição encontrada no *log* do servidor e determine o endereço de origem **responsável pelo maior número de requisições**. Escreva essa função em PHP.

Solução

```
function determinaOrigemMaisComum($acessos) {
  $contagem = array();
  $tamanho = count($acessos);
  $maximo = 0;
  $maisComum = "";

  for ($i = 0; $i < $tamanho; $i++) {
    if (isset($contagem[$acessos[$i]]))
      $contagem[$acessos[$i]] = $contagem[$acessos[$i]] + 1;
    else
      $contagem[$acessos[$i]] = 1;
    if ($contagem[$acessos[$i]] > $maximo) {
      $maximo = $contagem[$acessos[$i]];
```

```
      $maisComum = $acessos[$i];
    }
  }
  return($maisComum);
}
```

Explicação: a solução utiliza a capacidade do PHP de arrays associativos, ou seja, a possibilidade de usar *strings* como índices de vetores. Isso simplifica bastante o processo de contagem da frequência de cada endereço no vetor $acessos recebido como parâmetro.

A função varre o vetor $acessos. Para cada elemento (endereço de um acesso), verifica-se se já existe uma entrada correspondente no vetor $contagem. Em caso negativo, cria-se uma nova entrada em $contagem, inicializada com 1. Em caso positivo, incrementa-se a entrada correspondente.

Nessa mesma repetição, após cada atualização do vetor $contagem, verifica-se se a variável $maisComum (que armazena o endereço mais comum até então) deve ser atualizada. Isso é feito com base no valor da variável $maximo, que armazena o número de ocorrências do endereço $maisComum.

Observação: outras soluções corretas são possíveis.

Questão 4: Imagine que você trabalha em uma empresa especializada em desenvolvimento de jogos *on-line*. Em particular, seu projeto atual é o desenvolvimento de um simples jogo da velha em PHP. Como decisão de projeto, o tabuleiro de uma partida é armazenado na forma de uma matriz (isto é, um *array* bidimensional). Cada posição da matriz corresponde ao estado daquela casa do tabuleiro, podendo assumir como valores os caracteres ‘X’ (casa preenchida pelo jogador 1), ‘O’ (casa preenchida pelo jogador 2) ou ‘ ‘ (casa ainda não preenchida). Escreva uma função em PHP que receba como parâmetro a matriz do tabuleiro e imprima as seguintes informações:

- Se a partida acabou.
- Se houve vencedor ou empate (caso a partida tenha acabado).
- Qual foi o vencedor (caso haja um).

Lembre-se: a partida acaba quando há um vencedor ou quando todas as casas foram preenchidas. Vence a partida o jogador que conseguir preencher três casas subsequentes (isto é, na mesma linha, coluna ou diagonal). O empate ocorre quando todas as casas foram preenchidas, mas não há vencedor.

Solução

```
function fimDeJogo($tabuleiro) {
    for ($i = 0; $i < 3; $i++) {
        if ($tabuleiro[$i][0] == $tabuleiro[$i][1] && $tabuleiro[$i][0] == $tabuleiro[$i][2]) {
            if ($tabuleiro[$i][0] == 'X') {
                echo "Fim de partida. Jogador 1 venceu.\n";
                return ;
            }
            else if ($tabuleiro[$i][0] == 'O') {
                echo "Fim de partida. Jogador 2 venceu.\n";
                return ;
            }
        }
        if ($tabuleiro[0][$i] == $tabuleiro[1][$i] && $tabuleiro[0][$i] == $tabuleiro[2][$i]) {
            if ($tabuleiro[0][$i] == 'X') {
                echo "Fim de partida. Jogador 1 venceu.\n";
                return ;
            }
            else if ($tabuleiro[0][$i] == 'O') {
                echo "Fim de partida. Jogador 2 venceu.\n";
                return ;
            }
        }
    }

    if ($tabuleiro[0][0] == $tabuleiro[1][1] && $tabuleiro[0][0] == $tabuleiro[2][2]) {
        if ($tabuleiro[0][0] == 'X') {
            echo "Fim de partida. Jogador 1 venceu.\n";
            return ;
        }
        else if ($tabuleiro[0][0] == 'O') {
            echo "Fim de partida. Jogador 2 venceu.\n";
            return ;
        }
    }

    if ($tabuleiro[2][0] == $tabuleiro[1][1] && $tabuleiro[2][0] == $tabuleiro[0][2]) {
```

```
        if ($tabuleiro[2][0] == 'X') {
                echo "Fim de partida. Jogador 1 venceu.\n";
                return ;
        }
        else if ($tabuleiro[2][0] == 'O') {
                echo "Fim de partida. Jogador 2 venceu.\n";
                return ;
        }
    }

    for ($i = 0; $i < 3; $i++) {
        for ($j = 0; $j < 3; $j++) {
                if ($tabuleiro[$i][$j] == ' ') {
                        echo "Jogo ainda nao acabou.\n";
                        return ;
                }
        }
    }
    echo "Jogo acabou. Empate.\n";
}
```

Explicação:
A primeira repetição verifica se um dos dois jogadores conseguiu posicionar suas jogadas (caracteres ‘X’ ou ‘O’) em todas as posições de uma linha ou coluna do tabuleiro. Para isso, verifica-se para cada linha e coluna se seu conteúdo são três caracteres iguais. Se sim, verifica-se se o caractere em questão é ‘X’ ou ‘O’. Nesses casos, a função imprime que o jogo acabou e declara o respectivo jogador como ganhador.

Os dois blocos condicionais seguintes à primeira repetição testam se um dos dois jogadores foi capaz de preencher uma das diagonais do tabuleiro. As verificações executadas são similares às da primeira repetição.

Caso a execução chegue até a última repetição, não há (ainda, ao menos) vencedor para a partida. Esta última repetição, portanto, se encarrega de verificar se há ao menos uma posição do tabuleiro ocupada pelo caractere ‘ ‘, indicando que ainda há jogadas a serem feitas. Nesse caso, imprime-se que o jogo ainda não terminou.

Por fim, se a função continua sua execução após a última repetição, então todas as casas do tabuleiro foram preenchidas e não houve vencedor. Nesse caso, declara-se empate.

Observação: outras soluções corretas são possíveis.

Questão 5: Conforme visto na disciplina, a linguagem PHP permite programação orientada a objetos. Suponha que você esteja desenvolvendo um site que exiba o placar atualizado dos jogos do Campeonato Brasileiro de futebol. Escreva uma classe em PHP que represente uma partida. Essa classe deverá incluir:

- O nome dos times que estão jogando.
- O placar atual.
- A lista dos jogadores que marcaram gol para o time da casa.
- A lista dos jogadores que marcaram gol para o time visitante.
- O minuto da partida em que cada gol ocorreu.
- Um método para atualizar o placar e demais informações da classe quando ocorre um gol do time da casa.
- Um método para atualizar o placar e demais informações da classe quando ocorre um gol do time visitante.

Outras informações/métodos não são necessários.

Solução

```
class Partida {

   private $timeCasa = "";
   private $timeVisitante = "";
   private $timeCasaPlacar = 0;
   private $timeVisitantePlacar = 0;
   private $timeCasaGolsJogador = array();
   private $timeVisitanteGolsJogador = array();
   private $timeCasaGolsTempo = array();
   private $timeVisitanteGolsTempo = array();

   public function golTimeCasa($jogador, $tempo) {

         $this->timeCasaPlacar = $this->timeCasaPlacar + 1;
         $this->timeCasaGolsJogador[] = $jogador;
         $this->timeCasaGolsTempo[] = $tempo;
```

```
    }

    public function golTimeVisitante($jogador, $tempo) {

        $this->timeVisitantePlacar = $this->timeVisitantePlacar + 1;
        $this->timeVisitanteGolsJogador[] = $jogador;
        $this->timeVisitanteGolsTempo[] = $tempo;
    }
}
```

Explicação: Esta questão avalia o conhecimento da sintaxe dos elementos básicos da declaração de uma classe em PHP. Na solução proposta, é declarada uma classe chamada “Partida”. Começa-se pela declaração dos atributos da classe, conforme requisitados no enunciado (repare que não se exige a declaração dos atributos como privados, mas é importante a inicialização dos mesmos).

Em seguida, declaram-se os dois métodos solicitados pelo enunciado. Os métodos realizam manipulações básicas das propriedades da classe através da palavra-chave $this.