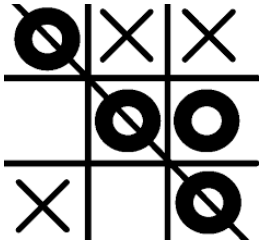
| | | |
|---|---|---|
| Q1 | 2,0 | |
| Q2 | 2,0 | |
| Q3 | 3,0 | |
| Q4 | 3,0 | |

**Fundação CECIERJ – Vice Presidência de Educação Superior à Distância**
**Curso de Tecnologia em Sistemas de Computação**
**Disciplina: Programação de Aplicações Web**
**Professores: Flávio Seixas e Marcos Lage**
**AP1 – 2º Semestre de 2019 - Gabarito**
**Nome:**_____

**Questão 1.** O Jogo da Velha é um jogo de regras extremamente simples. O tabuleiro é uma matriz de três linhas e três colunas. Dois jogadores escolhem uma marcação, podendo ser Xis (X) ou Círculo (O). Os jogadores jogam alternadamente, assinalando uma marcação por vez em uma célula que esteja vazia. O jogador vence se conseguir completar com três Xis ou Círculos em uma linha horizontal ou vertical ou diagonal. Escreva uma função em PHP que receba como parâmetro uma matriz 3 x 3, com as marcações -1 para célula vazia, 0 para Círculo e 1 para Xis. A função deve avaliar a distribuição das marcações na matriz e retornar -1 se não há vencedor, 0 se o vencedor foi o jogador da marcação Círculo, e 1 se o vencedor foi o jogador da marcação Xis.



```php
function verificaGanhador(array $tabuleiro) {

    // verifica linhas
    for($i = 0; $i < 3; $i++) {
        $c = 0;
        for($j = 0; $j < 3; $j++) {
```

```
            $c += $tabuleiro[$i][$j];
        }
        $soma[] = $c;
    }

    // verifica colunas
    for($j = 0; $j < 3; $j++) {
        $c = 0;
        for($i = 0; $i < 3; $i++) {
            $c += $tabuleiro[$i][$j];
        }
        $soma[] = $c;
    }

    // verifica diagonal principal
    $c = 0;
    for($i = 0; $i < 3; $i++) {
        $c += $tabuleiro[$i][$i];
    }
    $soma[] = $c;

    // verifica diagonal alternativa
    $c = 0;
    for($i = 0; $i < 3; $i++) {
        $c += $tabuleiro[$i][2-$i];
    }
    $soma[] = $c;

    if (in_array(0, $soma)) {
        return 0;   // o jogador com marcação Círculo venceu!
    }

    if (in_array(9, $soma)) {
        return 1;   // o jogador com marcação Xis venceu!
    }

    return -1;      // ainda não houve vencedor.
}

$tabuleiro = array(array(0, -1, 3), array(-1, 3, 3), array(3, 3, 0));
echo "\n".verificaGanhador($tabuleiro)."\n";
```

**Questão 2.** Utilizando o operador módulo (ex., $a % $b), escreva uma função em PHP que receba como parâmetro um número inteiro, e retorne **true** caso o número seja primo, **false** caso contrário.

```php
function verificaNumeroPrimo($numero) {

    if ($numero <= 1)
        return false;

    $d = $numero - 1;
    while($d > 1) {
        if ($numero % $d == 0) {
            return false;
        }
        $d--;
    }
    return true;
}
```

**Questão 3.** Escreva uma função em PHP que receba como argumento um vetor de números inteiros e potencialmente repetidos, e exiba os mesmos valores, mas sem repetição, seguido de quantas vezes o valor numérico aparece repetido no vetor, separado por “:”. Por exemplo, para um vetor com os valores 1, 7, 3, 1, 3, 1, sua função deve exibir os valores 1:3, 7:1, 3:2 (a ordem não é importante). A função deve retornar o número que apareceu mais vezes.

```php
function contaValores(array $valores) {
    $valores_unico = array_unique($valores);

    foreach($valores_unico as $v_unico) {
        $c = 0;
        foreach($valores as $v) {
            if ($v == $v_unico) {
                $c++;
            }
        }
        $conta[] = $c;
    }

    for ($i = 0; $i < sizeof($valores_unico); $i++) {
        echo $valores_unico[$i].":".$conta[$i]."\n";
    }
}
```

**Questão 4.** Considerando o suporte a linguagem PHP a orientação a objetos, deseja-se modelar uma classe **Turma**. Na classe Turma é composta pelos seguintes atributos:

- 01 vetor alunos, contendo de objetos Aluno;
- 01 atributo professor para o objeto Professor;

- 01 atributo disciplina para o objeto Disciplina.

A classe Turma implementa os métodos **vincularProfessor**, **vincularDisciplina** e **vincularAluno**, com a responsabilidade de atribuir um Professor, uma Disciplina e adicionar ao vetor Alunos um objeto Aluno, respectivamente. Assegurar que o mesmo aluno não seja vinculado duplamente a turma. Para garantir que o mesmo aluno não seja adicionado duplamente ao vetor alunos, utilizar a matrícula do aluno como indexador do vetor alunos. Implementar também um método que retorne a quantidade de alunos na turma.

A classe Professor possui como atributo o nome do professor. A classe Disciplina possui como atributo o nome da disciplina. A classe Aluno possui como atributo o nome do aluno e a matrícula do aluno, representada por um sequencial numérico.

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

```php
class Aluno {
    private $nome;
    private $matricula;

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

    function obterMatricula() {
        return $this->matricula;
    }

    function obterNome() {
        return $this->nome;
    }
}

class Professor {
    private $nome;

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

class Disciplina {
    private $nome;
```

```php
    function __construct($nome) {
        $this->nome = $nome;
    }
}

class Turma {
    private $alunos = array();
    private $disciplina;
    private $professor;

    function vincularProfessor($professor) {
        $this->professor = $professor;
    }

    function vincularDisciplina($disciplina) {
        $this->disciplina = $disciplina;
    }

    function vincularAluno($aluno) {
        $this->alunos[$aluno->obterMatricula()]= $aluno->obterNome();
    }
}

$aluno1 = new Aluno("Flavio", 12345);
$aluno2 = new Aluno("Joao", 23456);
$disciplina = new Disciplina("Programação em Aplicações Web");
$professor = new Professor("Luiz", 323232);

$turma = new Turma;
$turma->vincularDisciplina($disciplina);
$turma->vincularProfessor($professor);
$turma->vincularAluno($aluno1);
$turma->vincularAluno($aluno2);

var_dump($turma);
```