1)	**2,0**	
2)	**3,0**	
3)	**5,0**	

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
Gabarito da AD2 – 2º Semestre de 2018

Atualmente o jogo conhecido como Sudoku é muito popular. O objetivo do jogo é completar todas as casas de um tabuleiro de 9 por 9 utilizando números de 1 a 9.
Para completá-los, seguiremos a seguinte regra: Não podem haver números repetidos nas linhas horizontais nem nas colunas verticais, assim como em cada um dos 9 grupos 3 por 3 (veja a figura abaixo).

7	8	5	3	2	6	9	1	4
6	2	1	8	9	4	3	7	5
3	4	9	7	1	5	8	2	6
1	3	7	5	4	2	6	8	9
9	6	4	1	8	3	2	5	7
2	5	8	6	7	9	4	3	1
5	1	6	9	3	8	7	4	2
4	7	3	2	6	1	5	9	8
8	9	2	4	5	7	1	6	3

1) (2 pts) Implemente um programa em PHP para gerar tabuleiros Sudoku válidos (completamente preenchidos).

Dica: Este problema pode ser resolvido mais facilmente usando uma rotina recursiva. A rotina deve preencher uma casa com um dos possíveis candidatos, isto é, um dos numerous ainda não usados na coluna, na linha ou no quadrado grande. Se a casa não

pode ser preenchida por falta de candidatos, a rotina deve retornar FALSE. Caso haja mais de um candidato, cada tentativa de preenchimento deve ser sucedida de uma chamada recursive para preencher a casa seguinte. Se não há mais casas a preencher, a rotina retorna TRUE.

R:

```
define('N','9');
define('Q','3');

function imprime($matriz) {
     for($i = 0; $i < N; $i++) {
          for($j = 0; $j < N; $j++) {
               echo $matriz[$i][$j] . '   ';
          }
          echo '<br />';
     }
}

//condicao 1: nao haver repeticao de números em uma mesma linha
function validaLinha($matriz, $linha, $d) {
     for($j = 0; $j < N; $j++) {
          if($matriz[$linha][$j] == $d)
               return false;
     }
     return true;
}
//condicao 2: // por coluna
function validaColuna($matriz, $coluna, $d) {
     for($i = 0; $i < N; $i++) {
          if($matriz[$i][$coluna] == $d)
               return false;
     }
     return true;
}
//condicao 3: nao haver repeticao no mesmo quadrante
function validaBloco($matriz, $i, $j, $d) {
     $iInicial = (int) ($i / Q);
```

```
     $jInicial = (int) ($j / Q);
     for($i = 3*$iInicial; $i < Q + 3*$iInicial; $i++) {
          for($j = 3*$jInicial; $j < Q + 3*$jInicial; $j++) {
               if($matriz[$i][$j] == $d)
                    return false;
          }
     }
     return true;
}
// todas as condicoes
function valida($matriz, $i, $j, $d) {
     return validaLinha($matriz, $i, $d)
          && validaColuna($matriz, $j, $d)
          && validaBloco($matriz, $i, $j, $d);
}


//Backtracking Recursivo
//Recursivo, pois ora invoca a si mesma ora
//sinaliza true/false para quem a chamou
//Backtracking, pois retrocede para solucao parcial anterior
//e tenta de novo por outro caminho (tentativa e erro)
//1) Estou na Solucao?
//2) Nao.  Ok, onde posso ir?
//3) Vá lá.
//5) Achei a solucao?  se sim, return true!
//5) Existem mais lugares para ir, volte ao passo 1.
//6) Sem lugares para ir.  return false.

//Neste caso particular percorreremos da esquerda pra direita,
//de cima para baixo, quando acabarmos uma linha,
//vamos para proxima

function backtracking($i, $j) {
     global $tabuleiro, $encontrado;
```

```
//se atingimos a linha final, entao
//completamos o tabuleiro, podemos parar (estou na solucao)
if($i > N-1) {
     //imprime($tabuleiro);
     $encontrado = true;
     global $tabuleiro_inicializado;
     $tabuleiro_inicializado = $tabuleiro;
     return true;
}
if($encontrado)
     return false;

//se tabuleiro ja esta preenchido na posicao atual,
//seguimos para proxima (onde posso ir?)
if($tabuleiro[$i][$j] != null) {
     if($j >= N-1) {
          //nao da para seguir, pule a linha
          backtracking($i + 1, 0);
     } else {
          backtracking($i, $j + 1);
     }
}
//testamos os candidatos possiveis, em ordem aleatoria
$digitos = range(1,N);
shuffle($digitos);

foreach($digitos as $d) {
     if(valida($tabuleiro, $i, $j, $d)) {
          //aqui vamos entrar numa ramificacao
          //da arvore de possibilidades,
          //cosiderando que o digito aqual seja $d,
          //obs: pode dar errado, e teremos que voltar
          //apagando tudo que preenchemos para tentar outro
          $tabuleiro[$i][$j] = $d;

          //ramifica para os filhos,
```

```
                //vamos ver se retornam verdadeiro
                if($j >= N-1) {
                     //nao da para seguir, pule a linha
                     backtracking($i + 1, 0);
                } else {
                     backtracking($i, $j + 1);
                }
           }
     }

     //so vai chegar aqui se nenhum digito
     //for valido em nenhum caminho a partir do no pai atual.
     //esvaziaremos o valor desse no pai e retornaremos falso
     $tabuleiro[$i][$j] = null;
     return false;
}

function tabuleiroSudoku() {
     global $tabuleiro, $encontrado;

     $tabuleiro = [];
     for($i = 0; $i < N; $i++)
          for($j = 0; $j < N; $j++)
               $tabuleiro[$i][$j] = null;
     $encontrado = false;

     backtracking(0,0);

     return $tabuleiro;
}
```

2) (3 pts) Implemente um programa em PHP que recebe tabuleiros Sudoku válidos (totalmente preenchidos) como entrada, e remove desse tabuleiro dígitos que possam ser derivados dos remanescentes, removendo pelo menos um de cada fileira, coluna e bloco. As remoções devem manter a solução do jogo como única. A saída desse programa deverá ser um tabuleiro para ser jogado.

Observação: Quanto mais dígitos forem removidos, maior será a dificuldade do jogo.

R:

```
function inicializaSudoku() {
     global $tabuleiro_inicializado;
     $gabarito = $tabuleiro_inicializado;
     //17 é o numero minimo de digitos
     //Remover 9*9*9 posicoes = linhas, colunas, blocos
     $digitos = range(0,8);
     //por linha
     for($i = 0; $i < N; $i++) {
          shuffle($digitos);
          $tabuleiro_inicializado[$i][$digitos[0]] = null;
     }
     //por coluna
     for($j = 0; $j < N; $j++) {
          shuffle($digitos);
          $i = 0;
          while($tabuleiro_inicializado[$digitos[$i]][$j] ==
null) $i++;
          $tabuleiro_inicializado[$digitos[$i]][$j] = null;
     }
     //por bloco
     for($k = 0; $k < N; $k++) {
          $iInicial = ((int) ($k / Q)) * 3;
          $jInicial = ((int) ($k % Q)) * 3;
          shuffle($digitos);
          $i = -1;
          $dI = 0;
          $dJ = 0;
          do {
               $i++;
               $dI = (int) ($digitos[$i] / Q);
               $dJ = (int) ($digitos[$i] % Q);
          } while($tabuleiro_inicializado[$iInicial +
$dI][$jInicial + $dJ] == null);
          $tabuleiro_inicializado[$iInicial + $dI][$jInicial +
$dJ] = null;
     }

     /*PARA QUESTAO 3: $id =
inserirTabuleiro($tabuleiro_inicializado, $gabarito);*/

     return array($id, $tabuleiro_inicializado);
}
```

3) (5 pts) Construa uma página Web que apresente tabuleiros de Sudoku retornados pelo programa do item anterior, de modo que as posições vazias possam ser preenchidas por um jogador, após clique no botão inciar jogo.

Essa página deverá conter quatro botões, com finalidades distintas:

1. Iniciar – torna as posições vazias editáveis e inicia um contador de tempo;
2. Limpar – com a finalidade de apagar os valores de todas as posições editáveis;
3. Gerar novo tabuleiro – com finalidade de apresentar um novo tabuleiro;
4. Avaliar – compara a resposta gerada com a única solução possível. Caso o preenchimento do jogador não esteja correto, o mesmo deverá ser informado que a resposta encontra-se errada. Caso a resposta esteja correta, o jogador deverá ser informado. Além disso, o tabuleiro, seu gabarito, e o record (tempo do jogo atual) deverão ser armazenado em um banco de dados, caso o tabuleiro não tenha sido previamente armazenado (primeira vez que o tabuleiro é jogado). Se o tabuleiro já encontra-se armazendo no banco de dados, o campo record deverá ser atualizado, se for o caso.

R: Para esta questão, considere que os códigos da questão 1 e 2 estão inclusos, e que a linha comentada com os dizeres "PARA QUESTAO 3" esteja descomentada.

```
<?php

function conectaBD() {
     $servername = "localhost";
     $username = "root";
     $password = "";
     $dbname = "AD220182";

     // Create connection
     $conn = new mysqli($servername, $username, $password,
$dbname);
     // Check connection
     if ($conn->connect_error) {
          die("Falha: " . $conn->connect_error);
     }
     return $conn;
}

function inicializarBD() {
```

```
	$conn = conectaBD();

	// sql to create table
	$sql = "CREATE TABLE tabuleiros(
		id INTEGER AUTO_INCREMENT PRIMARY KEY,
		tabuleiro TEXT NOT NULL,
		gabarito TEXT NOT NULL,
		record INT NULL
	)";

	$conn->query($sql) or die($conn->connect_error);
	$conn->close();

}

function inserirTabuleiro($tabuleiro, $gabarito) {

	$conn = conectaBD();

	$tabuleiro_json = json_encode($tabuleiro);
	$gabarito_json = json_encode($gabarito);
	$sql = "SELECT id FROM tabuleiros WHERE tabuleiro =
'".$tabuleiro_json."'";

	$result = $conn->query($sql) or die($conn->connect_error);

	if($result->num_rows > 0) {
		$row = $result->fetch_assoc();
		$id = $row['id'];
	} else {
		$sql = "INSERT INTO tabuleiros(`id`, `tabuleiro`,
`gabarito`, `record`) VALUES (default, '".$tabuleiro_json."',
'".$gabarito_json."', -1)";
		$conn->query($sql) or die($conn->connect_error);
		$id = $conn->insert_id;
	}

	$conn->close();
	return $id;
}

function avaliar($id_tabuleiro, $candidato_gabarito,
$tempo_inicial, $tempo_final) {

	$conn = conectaBD();
```

```
     $candidato_gabarito_json =
json_encode($candidato_gabarito);

     $sql = "SELECT 1 FROM tabuleiros WHERE gabarito =
'".$candidato_gabarito_json."' AND id = " . $id_tabuleiro;

     $result = $conn->query($sql) or die($conn->connect_error);

     if($result->num_rows > 0) {
          $sql = "UPDATE tabuleiros SET record =
'".($tempo_final-$tempo_inicial)."' WHERE id = " . $id_tabuleiro;
          $conn->query($sql) or die($conn->connect_error);
          $conn->close();
          return true;
     } else {
          $conn->close();
          return false;
     }
}
?>
<html>
<head>
<style type='text/css'>
*{
     font-size:36px;
     text-align:center;
}
input[type=button],input[type=reset],input[type=submit] {
     padding: 12px;
}
input[type=text] {
     width: 100%;
     height: 100%;
     color:grey;
}
</style>
<script type='text/javascript'>
function limparInputs() {
     var elementos = document.getElementById('form').elements;
     for(var i = 0, elemento; elemento = elementos[i]; i++) {
          if(elemento.type == 'text') {
               document.getElementById('form').elements[i].value
= '';
          }
     }
}
```

```
function iniciar() {
     limparInputs();
     document.getElementById('contador').value = (new
Date).getTime();
     var elementos = document.getElementById('form').elements;
     for(var i = 0, elemento; elemento = elementos[i]; i++) {
          if(elemento.type == 'text') {

     document.getElementById('form').elements[i].disabled = '';
          }
     }
}

function avaliar() {
     document.getElementById('contador_fim').value = (new
Date).getTime();
     var elementos = document.getElementById('form').elements;
     for(var i = 0, elemento; elemento = elementos[i]; i++) {
          if(elemento.value == '') {
               window.alert('Voce precisa preencher todas as
lacunas.');
               return false;
          }
     }
     document.getElementById('form').submit();
     return true;
}
</script>
</head>
<body>
<form id="form" name="form" action="" method="POST">
<?php

if(isset($_POST['id_tabuleiro'])) {
     //echo($_POST['id_tabuleiro']); echo($_POST['matriz']);
echo($_POST['contador']); echo($_POST['contador_fim']);
     $gabarito_candidato = json_decode($_POST['matriz']);
     $dados = $_POST['dados'];
     for($i = 0; $i < N; $i++) {
          for($j = 0; $j < N; $j++) {
               if(!$gabarito_candidato[$i][$j]) {
                    $gabarito_candidato[$i][$j] = (int)
array_shift($dados);
               }
          }
     }
```

```
     $sucesso = avaliar($_POST['id_tabuleiro'],
$gabarito_candidato, $_POST['contador'], $_POST['contador_fim']);
     if($sucesso) {
          ?><script
type="text/javascript">window.alert('Parabens, seu record foi
gravado!');</script><?php
     } else {
          ?><script type="text/javascript">window.alert('Lamento,
ainda nao esta correto.');</script><?php
     }
}

if(!isset($_POST['matriz'])) {
     tabuleiroSudoku();
     list($id, $matriz) = inicializaSudoku();
     $contador = '';
} else {
     $id = $_POST['id_tabuleiro'];
     $matriz = json_decode($_POST['matriz']);
     $contador = $_POST['contador'];
}
echo '<input type=hidden name="matriz"
value="'.json_encode($matriz).'" />';
echo '<input type=hidden id="contador" name="contador"
value="'.$contador.'" />';
echo '<input type=hidden id="contador_fim" name="contador_fim"
value="" />';
echo '<input type=hidden id="id_tabuleiro" name="id_tabuleiro"
value="'.$id.'" />';

?>
<table width=630px height=630px align=center border=1>
<?php

$dados = null;
$disabled = 'disabled';
if(isset($_POST['dados'])) {
     $dados = $_POST['dados'];
     $disabled = '';
}
foreach($matriz as $linha) {
     echo '<tr>';
     foreach($linha as $elemento) {
          if(!$elemento) {
               $val = '';
               if(isset($dados)) $val = array_shift($dados);
```

```
                echo '<td width=10% height=10% ><input
name="dados[]" '.$disabled.' type=text value="'.$val.'" /></td>';
            }
            else {
                echo '<td width=10% height=10%
><b>'.$elemento.'</b></td>';
            }
        }
        echo '</tr>';
}

?>
</table>
<br />
<input type=button value="Iniciar"
onclick="javascript:iniciar();" ></input>
<input type=button value="Limpar"
onclick="javascript:limparInputs();"></input>
<input type=button value="Avaliar" onclick="javascript:return
avaliar();"></input>
<input type=button value="Gerar Novo"
onclick="javascript:window.location=window.location.href;"
></input>
</form>
</body>
</html>
```