Gabarito - AD2 de Programação II

1. (3 pontos) O código abaixo mostra o uso de uma classe DB, que encapsula a conexão a um Banco de dados, permite fazer consultas e mostrar os resultados da última consulta. Pede-se implementar a classe DB usando os conceitos de Programação Orientada a Objetos.

```
$db = new DB("server","admin","password","database");
$sql = "SELECT * FROM table";
$db->consulta($sql);
echo $db->mostrar_dados();
```

Obs.: No exemplo, "server" é o nome do servidor, "admin" é o username (login) de acesso ao banco de dados, "password" a senha, e "database" o nome do banco de dados. O método `mostrar_dados` retorna uma tabela formatada em html com duas colunas: a primeira com os nomes dos atributos da tabela e a segunda com o valor correspondente. Assim, o resultado do programa acima poderia ser exibido no navegador com o aspecto:

nome	Joao da Silva
endereco	Rua Bela, 220
telefone	22333344

Resp:

```
<?php

  class DB   {

    var $resultado;

    function DB($base, $server, $user, $pass) {
      $this->lastResult = NULL;
      mysql_connect($server, $user, $pass)
      or die('Server connexion not possible.');
      mysql_select_db($base)
      or die('Database connexion not possible.');
    }

    function consulta($consulta) {
      $this->resultado = mysql_query($consulta);
    }

    function mostrar_dados() {
      $result = $this->resultado;
      echo "<table border=\"1\" style=\"margin: 2px;\">".
           "<thead style=\"font-size: 80%\">";
      $numFields = mysql_num_fields($result);

      $tables    = array();
      $nbTables  = -1;
      $lastTable = "";
      $fields    = array();
```

```
      $nbFields  = -1;
      while ($column = mysql_fetch_field($result)) {
        if ($column->table != $lastTable) {
          $nbTables++;
          $tables[$nbTables] =
                  array("name" => $column->table, "count" => 1);
        } else
          $tables[$nbTables]["count"]++;
        $lastTable = $column->table;
        $nbFields++;
        $fields[$nbFields] = $column->name;
      }
      for ($i = 0; $i <= $nbTables; $i++)
        echo "<th colspan=".$tables[$i]["count"].
            ">".$tables[$i]["name"]."</th>";
      echo "</thead>";
      echo "<thead style=\"font-size: 80%\">";
      for ($i = 0; $i <= $nbFields; $i++)
        echo "<th>".$fields[$i]."</th>";
      echo "</thead>";

      while ($row = mysql_fetch_array($result)) {
        echo "<tr>";
        for ($i = 0; $i < $numFields; $i++)
          echo "<td>".htmlentities($row[$i])."</td>";
        echo "</tr>";
      }
      echo "</table></div>";
    }
  };

  $db = new DB("localhost", "root", "", "mysql");
  $db->consulta("select * from user")
  $db->mostrar_dados();
?>
```

2. (3 pontos) Faz algum tempo o jogo conhecido como **Sudoku** tornou-se 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). Pede-se implementar um programa em PHP para gerar um tabuleiro Sudoku válido.

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

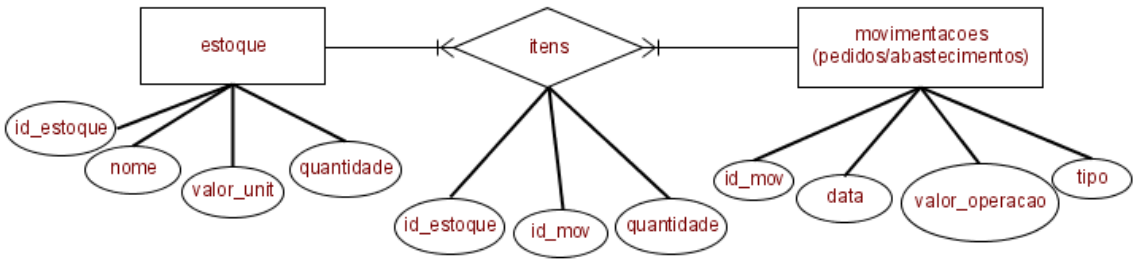
Dica: Este problema pode ser resolvido mais facilmente usando uma rotina recursiva. A rotina deve preencher uma casa com um dos possíveis candidatos, isto é, um dos números ainda não usados na coluna, na linha ou no quadrado grande. Se a casa não pode ser preenchida por falta de candidatos, a rotina deve retornar FALSE. Caso haja mais de um candidato, cada tentativa de preenchimento deve ser sucedida de uma chamada recursiva para preencher a casa seguinte. Se não há mais casas a preencher, a rotina retorna TRUE.

Resp:

```
<?php

   class sudoku{
      var $tab;
      var $ordem;
      var $vazios;
      var $vizinhanca;
      function sudoku(){
         // cria um tabuleiro vazio
         $this->tab = array_fill(0, 9, array_fill(0, 9, 0));
         for ($i = 0; $i<9*9; $i++)
            $this->ordem[$i] = array(floor($i/9), $i % 9 );
         //print_r($this->ordem);
         $this->vizinhanca = array();
         for ($i = 0; $i<9; $i++){
            $vizinhanca = array();
            for ($j = 0; $j<9; $j++)
               $vizinhanca[] = $this->_vizinhanca($i, $j);
            $this->vizinhanca[] = $vizinhanca;
         }
         $this->vazios = $this->ordem;
      }

      function _vizinhanca($i, $j){
         $vizinhanca = array();
         for ($k = 0; $k<9; $k++){
             if ($k != $j) $vizinhanca[] = array($i, $k);
             if ($k != $i) $vizinhanca[] = array($k, $j);
          }
          $k = floor($i/3)*3;
          $l = floor($j/3)*3;
          for ($ii = $k; $ii < $k+3; $ii++)
            for ($jj = $l; $jj < $l+3; $jj++)
               if ($i != $ii && $j != $jj)
                  $vizinhanca[] = array($ii, $jj);

         return $vizinhanca;
      }

      function _shuffle_assoc(&$array) {
         //$keys needs to be maintained
         if (count($array)>1) {
            $keys = array_rand($array, count($array));

            foreach($keys as $key)
              $new[$key] = $array[$key];

            $array = $new;
         }
      }

      function _candidatos($i, $j){
         $cand = array();
         for($k = 1; $k<10; $k++)
            $cand[$k] = $k;
         foreach($this->vizinhanca[$i][$j] as $cell){
            $val = $this->tab[$cell[0]][$cell[1]];
```

```
            $key = array_search($val, $cand);
            if (!empty($key)) unset($cand[$key]);
         }
         $this->_shuffle_assoc($cand);
         return $cand;
      }

      function _resolve(){
         if (empty($this->vazios)) return true;
         $cell = array_pop($this->vazios);
         foreach ($this->_candidatos($cell[0], $cell[1]) as $c ){
            $this->tab[$cell[0]][$cell[1]] = $c;
            if ($this->_resolve()) return true;
         }
         $this->tab[$cell[0]][$cell[1]] = 0;
         array_push($this->vazios, array($cell[0], $cell[1]));
         return false;
      }

      function _show(){
         echo "+-------+-------+-------+\n";
         for($i = 0; $i<9; $i++){
            echo "| ";
            for($j = 0; $j<9; $j++){
               echo $this->tab[$i][$j]." ";
               if (($j + 1) % 3 == 0) echo "| ";
            }
            echo "\n";
            if (($i + 1) % 3  == 0)
               echo "+-------+-------+-------+\n";
         }
      }

   };

   $sudoku = new sudoku;
   $sudoku->_resolve();
   $sudoku->_show();

?>
```

3. (4 pontos) Na primeira avaliação à distância foi pedida a modelagem completa de um banco de dados para um sistema de estoque. Utilizando a modelagem do gabarito (veja abaixo), escreva os comandos SQL necessários para:
   i. Mostrar o total de vendas de um determinado dia.
   ii. Mostrar os produtos vendidos num determinado dia.
   iii. Mostrar o saldo de um determinado mês.
   iv. Mostrar os produtos que precisam ser abastecidos, isto é, produtos com menos de 5 unidades no estoque.



Resp:

```
SELECT sum(valor_operacao) FROM movimentacoes
WHERE data LIKE '%2008-04-04%' AND t  ipo = 'V';

SELECT e.nome, sum(i.quantidade)
FROM movimentacoes as m, estoque as e, itens as i
WHERE m.data LIKE '%2008-04-04%' AND
      (i.id_estoque = e.id_estoque AND i.id_mov = m.id_mov) AND
      m.tipo = 'V' GROUP BY nome;

SELECT sum(valor_operacao), tipo FROM movimentacoes
WHERE (data > '2008-04-01' AND data < '2008-05-01')
GROUP BY tipo;

SELECT nome, quantidade FROM estoque
WHERE quantidade < 10;
```