Fundação CECIERJ – Vice Presidência de Educação Superior à Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação II
Gabarito da AD1 – 2º Semestre de 2012

Questão 1 (2 pontos)

Escolha (V)erdadeiro ou (F)also para as afirmações abaixo:

1. (F) Para acessar uma página web que foi escrita usando PHP, é necessário utilizar um navegador com plugin PHP.

2. (F) Se uma página contém o comando PHP `echo $s`, então, $s necessariamente tem que ser uma variável string.

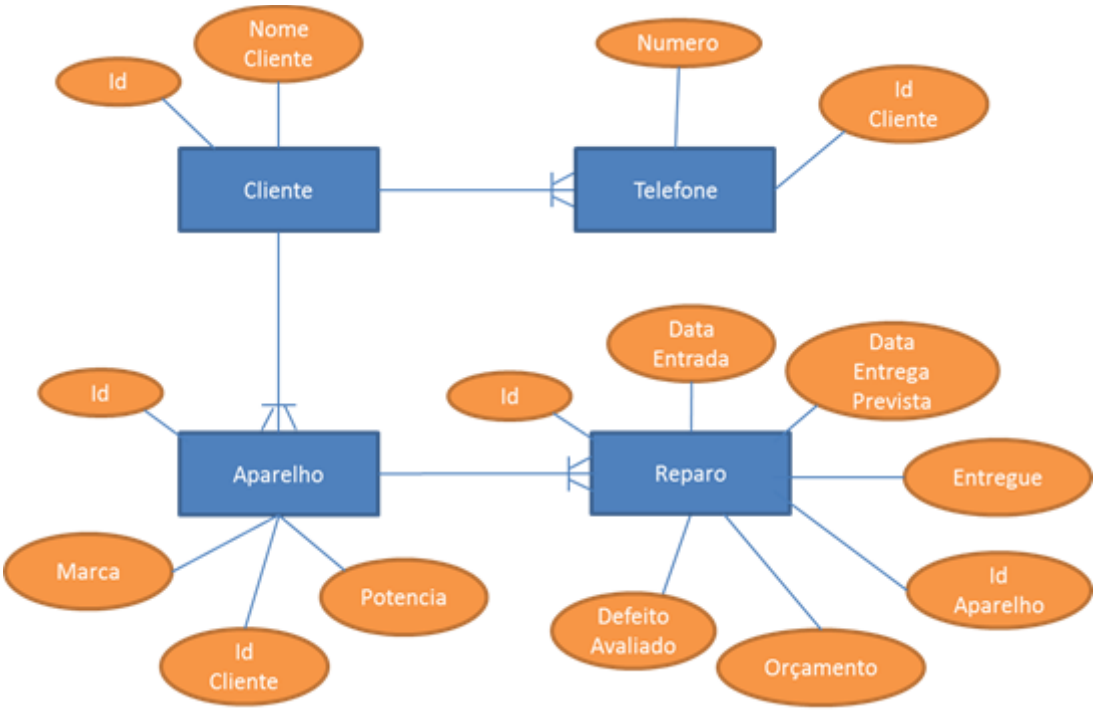
3. (F) A notação `$x++` incrementa a variável `$x` e é equivalente a escrever `++$x`.

4. (V) Se uma variável `$x` foi declarada fora de uma função, ela pode ser acessada de dentro da função desde que seja declarada num comando da forma `global $x`.

Questão 2 (1 ponto extra)

Uma firma de reparos de aparelhos de ar-condicionado precisa de um banco de dados para gerenciar os aparelhos deixados para conserto. Para cada aparelho deixado para conserto deve ser possível recuperar o cliente que o deixou, a data em que foi deixado, a potência em BTUs, e a marca do aparelho. Após o técnico encarregado examinar o aparelho, ele registra no banco de dados a data prevista para que aparelho fique pronto, uma descrição do defeito e o orçamento. Sempre que o aparelho é buscado pelo cliente, a entrada no banco de dados correspondente ao conserto é marcada como "entregue". O cliente tem que ter seu nome, endereço e telefones (mais do que um) registrados e deve ser possível recuperar do banco de dados os serviços ainda em aberto (não entregues) deixados pelo cliente. Pede-se: faça uma modelagem desse banco de dados, desenhe um modelo de entidades e relacionamentos correspondente e descreva como criar as tabelas do mesmo (modelo físico).

Resposta:




```
CREATE TABLE Cliente (

	Id INT UNSIGNED NOT NULL AUTO_INCREMENT,
	NomeCliente VARCHAR(255) NOT NULL,
	PRIMARY KEY(`Id`)
);
CREATE TABLE Telefone (

	IdCliente INT UNSIGNED NOT NULL,
	Numero VARCHAR(20) NOT NULL,
	FOREIGN KEY (`IdCliente`) REFERENCES Cliente(`Id`)
);
CREATE TABLE Aparelho (

	Id INT UNSIGNED NOT NULL AUTO_INCREMENT,
	Marca VARCHAR(255) NOT NULL,
	IdCliente int UNSIGNED NOT NULL,
	Potencia FLOAT NULL,
	PRIMARY KEY(`Id`),
	FOREIGN KEY (`IdCliente`) REFERENCES Cliente(`Id`)
);
CREATE TABLE Reparo (

	Id INT UNSIGNED NOT NULL AUTO_INCREMENT,
	DataEntrada DATETIME NOT NULL,
	DataEntregaPrevista DATETIME NOT NULL,
	DefeitoAvaliado VARCHAR(255),
	Orcamento DECIMAL(10,2) NOT NULL,
	Entregue TINYINT(1) DEFAULT 0,
	IdAparelho INT UNSIGNED NOT NULL,
```

```
	PRIMARY KEY(`Id`),
	FOREIGN KEY (`IdAparelho`) REFERENCES Aparelho(`Id`)
);
```

Questão 3 (4 pontos)

No jogo de pôquer tradicional, cada jogador recebe inicialmente uma mão com cinco cartas retiradas de um baralho convencional de 52 cartas. Dependendo da combinação de cartas, a mão vencedora é determinada de acordo com a seguinte tabela, em ordem decrescente de importância:

1. Straight flush: as cinco cartas em sequência e todas do mesmo naipe.
2. Four: quatro cartas de mesmo valor.
3. Full house: três cartas de mesmo valor e as demais duas cartas também de mesmo valor.
4. Flush: cinco cartas com mesmo naipe
5. Sequência: cinco cartas com valores em sequência.
6. Trinca: três cartas de mesmo valor.
7. Par: duas cartas de mesmo valor.
8. Nada: nenhuma combinação em particular.

Escreva uma função chamada `poquer()` que seleciona 5 cartas aleatoriamente de um baralho, as imprime e classifica a mão segundo a tabela acima. Por exemplo, ao chamar a função, o seguinte resultado poderia ser impresso:

```
10 de Copas,
Valete de Copas,
Dama de Copas,
Rei de Copas,
As de Copas: Straight Flush
```

Resposta:

```
<?php

$valores = array("As",2,3,4,5,6,7,8,9,10,"Valete","Dama","Rei");
$naipes = array(" de Paus"," de Ouros", " de Copas", " de Espadas");

function sorteioAleatorio() {
	$valor = rand(0, 12);
	$naipe = rand(0, 3);
	return array($valor, $naipe);
}

function poquer() {
	global $valores, $naipes;
	$valores_sorteados = array();
	$naipes_sorteados = array();
	$disponiveis = array_fill(0, count($valores), array_fill(0, count($naipes),
true));

	for($i = 0; $i < 5; $i++) {
		//OBS: sintaxe elegante nao será cobrada na correção
		list($valores_sorteados[$i],$naipes_sorteados[$i]) = sorteioAleatorio();
		//Se a carta sorteada não estiver disponível, refaça a iteração.
		if($disponiveis[$valores_sorteados[$i]][$naipes_sorteados[$i]]) {
```

```
            $disponiveis[$valores_sorteados[$i]][$naipes_sorteados[$i]] = false;
        } else {
            $i--;
            continue;
        }
        echo $valores[$valores_sorteados[$i]] . $naipes[$naipes_sorteados[$i]] .
($i != 4 ? ',<br/>' : '');
}

//Variável útil nos testes, mas deve ser usada com cuidado pois
// fere a ordem dos naipes
$valores_ordenados = $valores_sorteados;
sort($valores_ordenados);
$valores_distintos = array_values(array_unique($valores_sorteados));

//Provavelmente a grande maioria dos alunos fará loops com variáveis de teste
//ou ifs explicitos para todas as posicoes dos 2 arrays, o que dará diferente
//do gabarito, mas não há problema, desde que esteja correto.

//Straight Flush: mesmo naipe (naipes unicos = 1), e se ordenados,
//os valores sao todos diferentes, sendo o ultimo igual ao primeiro + 4
if(count(array_unique($naipes_sorteados)) == 1
&& count($valores_distintos) == 5
&& $valores_ordenados[0] + 4 == $valores_ordenados[4]) {
        echo ": Straight Flush";
        return;
}

//Four: 2 valores distintos, e se ordenados, o primeiro difere do segundo
//ou o penúltimo difere do último
if(count($valores_distintos) == 2
&& ($valores_ordenados[0] != $valores_ordenados[1]
|| $valores_ordenados[3] != $valores_ordenados[4])) {
        echo ": Four";
        return;
}
//Full House: 2 valores distintos, e se ordenados, o segundo difere
//do terceiro ou o terceiro difere do quarto
if(count($valores_distintos) == 2
&& ($valores_ordenados[1] != $valores_ordenados[2]
|| $valores_ordenados[2] != $valores_ordenados[3])) {
        echo ": Full House";
        return;
}

//Flush
if(count(array_unique($naipes_sorteados)) == 1) {
        echo ": Flush";
        return;
}

//Sequência
if(count($valores_distintos) == 5
&& $valores_ordenados[0] + 4 == $valores_ordenados[4]) {
        echo ": Sequencia";
        return;
```

```
        }

        //Trinca: Pode ser 2 pares e 1 distinto ou uma trinca e 2 distintos.
        //Então basta encontrar 3 chaves para um mesmo valor no array
        //de valores sorteados
        if(count($valores_distintos) == 3)
            for($i = 0; $i < 3; $i++)
               if(count(array_keys($valores_sorteados,$valores_distintos[$i])) == 3) {
                    echo ": Trinca";
                    return;
               }

        //Par: Se são 4 valores distintos então 1 valor é repetido = par
        //Se são 3 valores distintos: os casos que são trinca já foram tratados
        //Restando os casos em que temos 2 pares e 1 distinto
        if(count($valores_distintos) == 4
        || count($valores_distintos) == 3) {
              echo ": Par";
              return;
        }
        //Nada
        echo ": Nada";
}

poquer();

?>
```

Questão 4 (4 pontos)

Escreva a função `somafracoes($num1,$den1,$num2,$den2)` que imprime o resultado da soma de duas frações, a saber `$num1 / $den1 + $num2 / $den2`. Os argumentos da função bem como o resultado da função são frações usando números inteiros como numerador e denominador. Na fração resultante o numerador e o denominador devem ser primos entre si, ou seja a fração deve ser a mais simples possível. Por exemplo: `somafracoes(1,2,3,4)` deve escrever:

```
1 / 2 + 3 / 4 = 5 / 4
```

Resposta:

```
<?php

function somafracoes($num1,$den1,$num2,$den2) {
      echo $num1 .' / '. $den1 .  ' + ' . $num2 . ' / ' . $den2 . ' = ';
      $den_final = mmc($den1, $den2);
      $num_final = $num1 * $den_final / $den1
                              + $num2 * $den_final / $den2;

      while(mdc($num_final, $den_final) != 1) {
            $fator = mdc($num_final, $den_final);
            $den_final /= $fator;
            $num_final /= $fator;
      }

      echo $num_final . ' / ' . $den_final;
```

```
}

function mdc($n, $m) {
      $n=abs($n);
      $m=abs($m);
      if ($n==0 and $m==0)
         return 1; //divisao por zero
      if ($n==$m and $n>=1)
         return $n;
      return $m < $n ? mdc($n-$m,$n) : mdc($n,$m-$n);
}

function mmc($n, $m) {
   return $m * ($n/mdc($n,$m));
}

somafracoes(1, 2, 3, 4);

  ?>
```