Fundação CECIERJ – Vice Presidência de Educação Superior à Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação II
AP1 – 1º Semestre de 2013 - GABARITO

Questão 1

Escreva uma função chamada **edicao** que recebe duas *strings* de mesmo comprimento e analisa o quão parecidas são as letras contidas em posições correspondentes nas duas *strings*. Utilize a seguinte regra de pontuação para comparar cada letra da primeira string com a letra correspondente na segunda string:

Caracteres	Pontos
Letras iguais	0
Mesma letra, com diferença de maiúscula/minúscula	1
Letras diferentes	3
Um caractere não letra versus uma letra	10
Outros caracteres (não letras)	0

Assim, caso as duas strings sejam idênticas, sua função deve retornar 0. Veja os seguintes exemplos:

edicao("PHP", "PHP") ----> ***retorna 0***
edicao("pHp", "PHP") ----> ***retorna 2***
edicao("jhg", "PHP") ----> ***retorna 7***
edicao("P P", "PHP") ----> ***retorna 10***
edicao("PHP!", "PHP?") ----> ***retorna 0***

Gabarito - Questão 1

```
function edicao($str1, $str2) {
  $retorno = 0; //o contador inicia em zero.
  for($i = 0; i < strlen($str1); $i++) {
  /*
   * As strings serão olhadas caractere a caractere. Como ambas tem o mesmo
   * tamanho, o laço pode terminar quando o índice atinge o tamaho da
   * primeira string.
   */
      if (eLetra($str1{$i}) && eLetra($str2{$i})) {
      /*
       * Entra nesse if quando a posição $i das suas strings contém letras,
       * satisfazendo as linhas 1, 2 e 3 da tabela dada na questão.
       */
          if ($str1{$i} == $str2{$i}) {
              //Acrescenta zero ao retorno. Ou, não faz nada.
          } else if
             (($str1{$i}>$str2{$i}) &&( $str1{$i}==($str2{$i} - 'a' + 'A'))||
              ($str2{$i}>$str1{$i}) && ($str2{$i}==($str1{$i} - 'a' + 'A'))){
               /*
                * Na tabela ASCII, primeiro aparecem as letras de 'A' a 'Z'
                * maiúsculas, depois das letras de 'a' a 'z' minúsculas.
                * A operação (- 'a' + 'A') sobre qualquer caractere minúsculo
                * transforma-o em maiúsculo. Se o resultado da comparação for
                * igual, então os caracteres são os mesmos, mas com caixas
                * diferentes.
                */

              $retorno += 1; //De acordo com a tabela.
          } else {
              //Se ambos são letras, mas não são o mesmo caractere...
              $retorno += 3; //De acordo com a tabela.
          }
      } else if (( eLetra($str1{$i}) && !eLetra($str2{$i})) ||
                  (!eLetra($str1{$i}) &&  eLetra($str2{$i}))) {

          //Se um deles é letra e o outro não.
          $retorno += 10; //De acordo com a tabela.
      }
      //Se nenhum dos dois for letra, não faz nada.
  }
  return $retorno;
}
```

Questão 1 - Parte 1

```php
function eLetra($char) {
    //Um caractere é letra se estiver entre 'a' e 'z' ou entre 'A' e 'Z'
    if(($char >= 'a' && $char <= 'z') ||
       ($char >= 'A' && $char <= 'Z')) {
        return true;
    } else {
    //Caso contrário, ele não é.
        return false;
    }
}
```

Questão 1 - Parte 2

Questão 2

Considere a classe que representa um produto de um estoque:

```
class Produto {
   public $id, $nome, $quant, $preco;
};
```

Na qual: $id representa o código de identificação; $nome, o nome do produto; $quant, a quantidade disponível no estoque; e $preco, o preço de venda.

a) escreva uma função que recebe um *array* de produtos e o ordena segundo o código de identificação.

b) Sem usar funções da biblioteca padrão, escreva uma função chamada **buscaProduto** que faz uma busca no *array* (assumindo que ele já está ordenado). Essa função recebe como parâmetros de entrada um ***array*** de produtos e um **código** de um produto que se deseja buscar. A função deve ter como valor de retorno a **quantidade** disponível no estoque desse produto, caso esteja presente no vetor, ou retornar **-1** caso não seja encontrado no vetor.

Considere a classe Estoque, na qual um *array* armazena os produtos presentes em um estoque de forma ordenada (ordem crescente) pelo código de identificação de cada produto.

c) Declare a classe Estoque contendo o *array* de Produtos e os métodos **buscaProduto** e **ordenaProdutos**, criados nos itens b e c.

Gabarito - Questão 2 - Item A

```php
function ordenaProdutos($arrayProdutos) {
    /*
     * Existem diferentes algoritmos de ordenação. Qualquer um deles serve
     * para resolver essa questão. O algoritmo escolhido aqui é conhecido
     * como Bubble Sort, pois sua implementação é a mais intuitiva que
     * existe.
     */
    $numProdutos = count($arrayProdutos);
    $arrayOrdenado = array();

    for($i = 0; $i < $numProdutos; $i++) {
        /*
         * Deseja-se criar um $arrayOrdenado com o mesmo número de posições
         * que o array dado como entrada. O algoritmo só para quando o array
         * novo contiver todos os elementos do array antigo.
         */

        $menor = 0;
        /*
         * Inicialmente, diz-se que a posição que contém o menor elemento
         * é a posição zero.
         */
        for ($j = 0; $j < count($arrayProdutos); $j++) {
             //Varre-se o array de produtos procurando o menor elemento.
            if ($arrayProdutos[$j]->id < $arrayProdutos[$menor]->id) {
                /*
                 * usando $menor como referencia, procura-se algum 'id'
                 * menor. Caso ele esja encontrado, a nova posição de
                 * referencia é $j.
                 *
                 * $arrayProdutos[$j]->id referencia a variável id dentro
                 * do objeto Produto, que é o atributo comparado para
                 * a ordenação.
                 */
                $menor = $j;
            }
        }
        $arrayOrdenado[] = $arrayProdutos[$menor];
        /*
         * Sabendo-se que que $menor é Produto que possui menor id,
         * coloca-o na próxima posição de $arrayOrdenado.
         */
        unset($arrayProdutos[$menor]);
        //Remove o elemento já ordenado do array desorndenado.
    }
    return $arrayOrdenado;// Retorna o array ordenado.
}
```

Gabarito - Questão 2 - Item B

```php
function buscaProduto($arrayOrdenado, $id) {
    for($i = 0; $i < count($arrayOrdenado); $i++) {
        if ($id == $arrayOrdenado[$i]->id)
            return $arrayOrdenado[$i]->quant;
        /*
         * Para cada Produto, verifica se seu id é igual ao
         * id procurado. Se for, retorna a quantidade do produto
         */
        else if ($id < $arrayOrdenado[$i]->id) return -1;
        /*
         * Sabendo que o array dado como entrada está ordenado,
         * a busca pode ser interrompida se o id procurado for
         * menor do que o id do Produto corrente.
         */
    }
    return -1;
    /*
     * Retorna -1 se id procurado não for encontrado.
     */
}
```

Gabarito - Questão 2 - Item C

```php
class Estoque { //Declaração da classe estoque
    public $arrProdutos; //Que contém um array de produtos

    /*
     * As funções se transformam em métodos e passam a
     * trabalhar com o atributo $arrProdutos, ao inves de
     * arrays passados como parâmetro.
     *
     * Todos os $arrayOrdenado são substituídos por
     * $this->arrProdutos e pequenas alterações são feitas a
     * fim de transformar as funções em métodos.
     */
    function ordenaProdutos() {
        $arrayProdutos = $this->arrProdutos;
        //$this->arrProdutos é guardado numa variável auxiliar

        $numProdutos = count($arrayProdutos);
        $this->arrProdutos = array();
        //$this->arrProdutos é reiniciado.
        for($i = 0; $i < $numProdutos; $i++) {
            $menor = 0;
            for ($j = 0; $j < count($arrayProdutos); $j++) {
                if ($arrayProdutos[$j]->id < $arrayProdutos[$menor]->id) {
                    $menor = $j;
                }
            }
            $this->arrProdutos[] = $arrayProdutos[$menor];
            unset($arrayProdutos[$menor]);
        }
        //$this->arrProdutos torna-se um array ordenado.
    }

    function buscaProduto($id) {
        for($i = 0; $i < count($this->arrProdutos); $i++) {
            if ($id == $this->arrProdutos[$i]->id)
                return $this->arrProdutos[$i]->quant;
            else if ($id > $this->arrProdutos[$i]->id)
                return -1;
        }
        return -1;
    }
}
```

Questão 3

Em uma empresa, deseja-se criar um Banco de Dados Relacional para gerenciar sua área de **Controle de Pedidos**. Uma equipe de avaliação decidiu a estrutura do banco deve ser modelada a fim de satisfazer os seguintes requisitos:

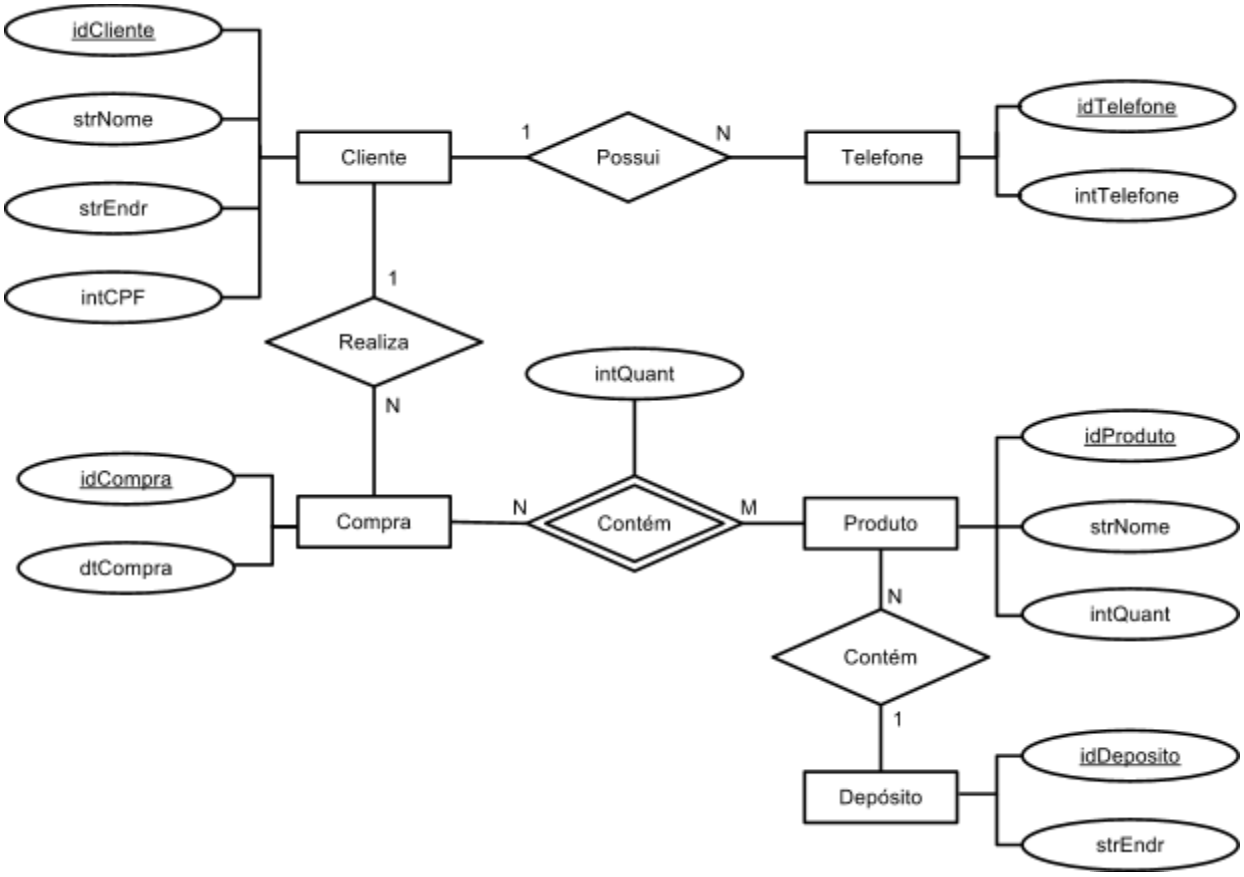
- Nome completo, endereço, CPF e telefone dos **clientes**. O sistema deve ser capaz de armazenar vários números de **telefone** para o mesmo cliente.
- Nome e quantidade de cada **produto** em estoque.
- A **compra** de um cliente é realizada à partir de pedidos por telefone. Um mesmo cliente pode pedir até vinte (20) produtos diferentes numa mesma compra, que será sempre entregue no endereço que consta no cadastro do cliente. Também é importante saber em qual dia o pedido foi realizado e qual quantidade de cada produto foi requisitada.
- Os produtos ficam armazenados em cinco (5) **estoques** diferentes, de maneira que, unidades de um mesmo produto fiquem sempre armazenadas no mesmo estoque. O sistema deve guardar o endereço do estoque, tal como seu número de identificação. Além de ser capaz de responder em qual estoque um produto se encontra, caso esteja disponível.

Com base nos requisitos apresentados, responda os itens abaixo.

a) Desenhe o Modelo Entidade-Relacionamento e identifique os atributos, as chaves, as restrições e os relacionamentos de cada uma das entidades relacionadas.

b) Escreva o comando SQL para criar a tabela **Cliente**, com seus atributos, chaves e restrições, de acordo com o diagrama apresentado no item a.

Gabarito - Questão 3 - Item A



Gabarito - Questão 3 - Item B

```
CREATE TABLE Cliente (
  idCliente INT NOT NULL AUTO_INCREMENT ,
  strNome VARCHAR(45) NOT NULL ,
  strEndr VARCHAR(45) NOT NULL ,
  intCPF INT NOT NULL ,
  PRIMARY KEY (idCliente)
);
```