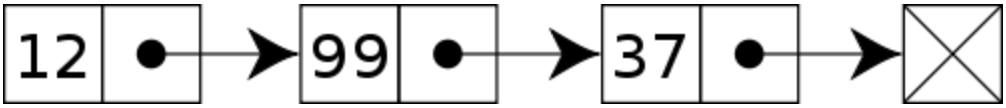
Fundação CECIERJ – Vice Presidência de Educação Superior à Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação II
Gabarito da AD1 – 1º Semestre de 2013

Questão 1

Considere que desejamos implementar uma lista encadeada composta por números inteiros **sem repetição** em PHP, utilizando como base um array **global** nativo da linguagem (chame este array de $lista). Estude seu funcionamento e implemente as seguintes funções de forma a prover todas as possibilidades listadas a seguir (é permitido uso de funções nativas da linguagem para simplificar a implementação):

- inserir($elemento) : mapeia o array global e insere um elemento na frente da lista.
- inserir_fim($elemento): insere o novo elemento como último elemento do array global.
- esta_vazia() : indica se a lista está vazia ou se possui algum elemento com um retorno booleano.
- remover() : remove o primeiro elemento da lista e retorna seu valor.
- remover_fim(): remove o último elemento da lista e retorna seu valor.
- buscar($elemento) : retorna a posição em que se encontra um elemento da lista (começando da posição 0).
- proximo($elemento) : retorna o valor do elemento na posição seguinte ao elemento buscado.
- anterior($elemento) : retorna o valor do elemento na posição anterior ao elemento buscado.

R: Atente para esta definição: “Uma lista encadeada é uma sequência de células na qual cada célula contém um objeto de algum tipo (no nosso caso números inteiros) e o endereço da célula seguinte”. Como em nossa descrição, o array não deve ter elementos repetidos, é preciso se preocupar com este aspecto durante a codificação.



No PHP não se tem disponível a possibilidade de manipulação explícita de ponteiros. A representação nesse exercício deve ser, portanto, uma aproximação da estrutura original, que expresse comportamento equivalente à uma implementação de mais baixo nível, por meio do resultado das chamadas de função definidas no exercício, aplicadas à um array comum.

```php
<?php

$lista = array();

function inserir($elemento)
{
    global $lista;
    if (is_int($elemento) && !in_array($elemento, $lista)) {
        array_unshift($lista, $elemento);
    } else die("Entrada invalida");
}

function inserir_fim($elemento)
{
    global $lista;
    if (is_int($elemento) && !in_array($elemento, $lista)) {
        $lista[] = $elemento; //ou array_push
    } else die("Entrada invalida");
}

function esta_vazia()
{
    global $lista;
    return empty($lista);
}

function remover()
{
    global $lista;
    if (!esta_vazia()) {
        return array_shift($lista);
    }
}

function remover_fim()
{
    global $lista;
    if (!esta_vazia()) {
```

```
        return array_pop($lista);
    }
}

function buscar($elemento)
{
    global $lista;
    if (is_int($elemento)) {
        return array_search($elemento, $lista);
    } else die("Parametro de busca invalido");
}

function proximo($elemento)
{
    global $lista;
    $tamanho_lista   = count($lista);
    $indice_elemento = buscar($elemento);
    if ($indice_elemento >= 0 && $indice_elemento < $tamanho_lista - 1)
        return $lista($indice_elemento + 1);
}

function anterior($elemento)
{
    global $lista;
    $indice_elemento = buscar($elemento);
    if ($indice_elemento > 0)
        return $lista($indice_elemento - 1);
}

?>
```

Questão 2

Escreva um programa que receba como entrada um ano compreendido entre 1900 e 2999 e responda qual a data do Carnaval para aquele ano. Dica: Algoritmo de Delambre apresenta melhor resultado.

R: Para calcular-se o carnaval, precisamos da data da páscoa:

```
function data_pascoa($ano) {

                $A = ($ano % 19);
                $B = (int)($ano / 100);
                $C = ($ano % 100);
                $D = (int)($B / 4);
                $E = ($B % 4);
                $F = (int)(($B + 8) / 25);
```

```
                $G = (int)(($B - $F + 1) / 3);
                $H = ((19 * $A + $B - $D - $G + 15) % 30);
                $I = (int)($C / 4);
                $K = ($C % 4);
                $L = ((32 + 2 * $E + 2 * $I - $H - $K) % 7);
                $M = (int)(($A + 11 * $H + 22 * $L) / 451);
                $mes = (int)(($H + $L - 7 * $M + 114) / 31);
                $dia = (($H + $L - 7 * $M + 114) % 31) + 1;

                //Anos maiores ou iguais a 2038 nao serao representados
                //com retorno date em computadores x86. Retornar string
                return $dia.'/'.$mes.'/'.$ano.'<br/>';
}

function bissexto($ano) {
    return $ano % 400 ? true : $ano % 100 ? false : $ano % 4 ? true : false;
}

function data_carnaval($ano) {
       $pedacos_ano=explode("/", data_pascoa($ano));

       if (bissexto($ano)) {
           //operacao sobre ano 2000 que foi bissexto, e menor que 2038
           $ano_equivalente = date('m/d/Y', mktime(0,0,0,$pedacos_ano[1],
                               $pedacos_ano[0]-47, 2000));

       } else {
           //operacao sobre um ano nao bissexto e menor que 2038
           $ano_equivalente = date('m/d/Y', mktime(0,0,0,$pedacos_ano[1],
                               $pedacos_ano[0]-47, 2001));
       }
       $pedacos_ano_equivalente = explode("/", $ano_equivalente);
       return $pedacos_ano_equivalente[1].'/'.$pedacos_ano_equivalente[0]
              .'/'.$ano.'<br/>';
}
```

Questão 3

Descreva o mecanismo com o qual PHP gerencia memória. Como ele funciona? Utilize também trechos de código para exemplificar sua explicação.

R: Todas as variáveis do PHP são armazenadas internamente em uma estrutura chamada _zval_struct ou simplesmente zval. Os campos desta estrutura são: o tipo interno da variável, seu valor e o índice para o contador de referências, que fica localizado em outra estrutura, chamada tabela de símbolos. O PHP usa contagem de referências, garbage collector e cópia na escrita para gerenciar memória. Quando um contador de referências chega a zero na tabela

de símbolos, sua linha é removida e o garbage collector pode ter certeza de que a memória pode ser liberada. Com a cópia na escrita, o PHP somente aloca espaço para uma nova variável que faça referência a uma variável já existente se houver escrita. Do contrário ele mantém internamente a referência à variável anterior, evitando uso desnecessário de memória.
Um exemplo de uma situação em que o garbage collector não consegue resolver é quando ocorre referência circular. No código abaixo, a variável pai possui uma referência da filha, ao passo em que a filha guarda também a referência para sua variável pai.

```php
<?php
class Foo {
    function __construct()
    {
        $this->bar = new Bar($this);
    }
}
class Bar {
    function __construct($foo = null)
    {
        $this->foo = $foo;
    }
}
while (true) {
    $foo = new Foo();
    unset($foo);
    echo number_format(memory_get_usage()) . "\\";
}

?>
```

Neste âmbito, quando não é possível remover a referência circular, o desenvolvedor deve manualmente chamar a função unset() de dentro de uma das classes, na forma, por exemplo, de um método destrutor.
A diretiva global se trata de um tipo especial de alocação de memória, onde se faz referência à um zval localizado no escopo global. Portanto é equivalente, em um escopo mais restrito do que o global, escrever:

```php
// usar a palavra reservada global para referenciar uma variavel global
global $example;

// criar a referencia diretamente para a variavel global
$example =& $GLOBALS['example'];
```

Questão 4

Supondo que deseja reutilizar funções PHP descritas em um servidor remoto no qual PHP está habilitado. O que aconteceria ao incluir em seu programa um arquivo remoto terminado em *.php* deste outro webserver (usando include ou require)?

R: Se a diretiva allow_url_include estiver habilitada, os comandos include ou require resultarão na inclusão da saída impressa pelo script remoto (via echo, print, etc.), e não pela inclusão do código PHP em si. Não é possível acessar o código-fonte PHP externo apenas usando protocolo HTTP. Uma alternativa, caso seja possível requisitar alterações no script remoto, é adaptá-lo para receber parametros via GET e em seguida tratar localmente seu retorno como uma string.