Q1	3,0	
Q2	3,0	
Q3	4,0	

Fundação CECIERJ – Vice Presidência de Educação Superior à Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação de Aplicações Web
Professores: Flávio L. Seixas e Miguel Elias M. Campista
AD1 – 1º Semestre de 2020

Esta AD avalia o uso das estruturas básicas de repetição e condição, a criação e uso de funções, manipulação de vetores e programação orientada a objetos na linguagem PHP.

1) (3,0 pontos) Em telecomunicações, paridade refere-se ao número de bits '1' de um determinado número binário. Para assinalar a paridade, é adicionado um dígito binário de paridade ao final de uma sequência binária, ou bloco. Existem dois tipos de paridade: a paridade par e a paridade ímpar. A paridade será par quando o número de bits de valor '1' do bloco for ímpar e, nesse caso, é adicionado um bit de valor '1' ao final do bloco, tornando o número de bits '1' par; caso contrário, será ímpar. A figura abaixo mostra um exemplo de cálculo de paridade par 'P' utilizando um bloco de 7 bits, A_1 a A_7.

A_1	A_2	A_3	A_4	A_5	A_6	A_7	P
1	1	0	1	0	0	1	0

Mostre a implementação de uma função em PHP de nome `calculoDaParidadePar()`. Esta função deve receber um bloco de 32 bits, e retornar o bit de paridade (tipo inteiro). Utilizar um `array` para expressar o bloco de 32 bits.

2) (3,0 pontos) Um cliente deseja ocultar todos os endereços de email de uma longa cadeia de caracteres. Escrever uma função PHP que receba uma cadeia de caracteres, e que retorne outra cadeia de caracteres, substituindo todos os endereços email por

`xxx`. Por exemplo, para a entrada "`Eu, Flávio fseixas@ic.uff.br, gostaria de solicitar o cadastro de Beltrano beltrano@hotmail.com e Sicrano sicrano@uol.com.br.`", a função deve retornar "`Eu, Flávio xxx, gostaria de solicitar o cadastro de Beltrano xxx e Sicrano xxx.`".

3) (4,0 pontos) Você foi contratado para implementar em PHP um software para ajudar um viajante. O software deve ser implementado utilizando os recursos de orientação a objetos. O viajante Fulano (classe `Viajante`) inicia uma viagem de *n* dias, com *x* valor em unidades monetárias do país destino (u.m.).

A linha abaixo mostra como a classe `Viajante` deve ser instanciada. No caso, o viajante `Fulano`, inicia uma viagem de 20 dias, com 1000 u.m.

```
$viajante = new Viajante("Fulano", 20, 1000);
```

O usuário deve então lançar os itens de despesas do dia (classes `Despesas` e `ItemDespesa`). As despesas terão os seguintes atributos:

- Tipo, podendo ser: (1) Deslocamento, (2) Bilhetes de entrada, (3) Alimentação, (4) Compras e (5) Outros.
- Número do dia. Considera-se '1' o primeiro dia de viagem, e '20' o último.
- Valor: em unidades monetárias do país destino.

A linha abaixo mostra como as classes `Despesas` e `ItemDespesa` devem ser instanciadas. No caso, o usuário está lançando uma despesa do tipo `Descolamento`, no primeiro dia de viagem, com o valor de 55 u.m.

```
$despesas = new Despesas($viajante);

$despesas->adicionar(new ItemDespesa(TipoDespesa::Deslocamento, 1, 55));
```

Escrever uma função que retorne a despesa diária projetada para os dias restantes da viagem. A despesa diária projetada será calculada pelo valor restante (valor inicial subtraído da soma das despesas lançadas) dividido pelo número de dias restantes (número de dias de viagem subtraído pelo maior número de dia já lançado nas despesas).

A despesa diária projetada deverá ser exibida em uma tabela HTML. Em uma outra tabela HTML, deverão ser exibidas as despesas totais de cada dia da viagem, uma linha para cada dia de viagem.