Questões para candidato ao programa de estágio incubadora ASP.NET Luby

> - Esse teste inclui 10 questões envolvendo lógica de programação e estrutura de dados.
> - 10 questões envolvendo criações de scripts SQL.
> - Desafio para criação de um pequeno aplicativo que replica um funcionamento de uma máquina de venda de bebidas em lata, utilizando os conceitos em orientação a objetos.
> - As 10 questões envolvendo lógica de programação e estrutura de dados e o desafio devem ser realizados preferencialmente com **C#**.
> - Pode ser utilizado qualquer função nativa da própria linguagem utilizada.
> - Não utilizar dependências ou bibliotecas externas.

Crie um repositório no github para envio do código com as repostas.

---

1 Lógica de Programação

- 1.1 Implemente a função abaixo para calcular fatorial de um número.

```
 // 5! = 5 · 4 · 3 · 2 · 1 = 120
CalcularFatorial(5) == 120//true
```

- 1.2 Implemente a função abaixo que calcula o valor total do prêmio somando fator do tipo do prêmio conforme valores:
  - Tipo: "basic" fator multiplicação do prêmio: 1
  - Tipo: "vip" fator multiplicação do prêmio: 1.2
  - Tipo: "premium" fator multiplicação do prêmio: 1.5
  - Tipo: "deluxe" fator multiplicação do prêmio: 1.8
  - Tipo: "special" fator multiplicação do prêmio: 2
- **Regras**
  - A função também deverá provir um parâmetro para que seja passado fator de multiplicação próprio.
  - Quando parâmetro de fator de multiplicação próprio for informado e válido o mesmo deve sobrescrever o cálculo do tipo de prêmio.
  - O prêmio nunca deve ter um valor negativo ou igual a zero.

```
 CalcularPremio(100, "vip", null) == 120.00;//true
CalcularPremio(100, "basic", 3) == 300.00;//true
```

- 1.3 Implemente a função abaixo para contar quantos números primos existe até o número informado.

```
 //Número primo: 2
//Número primo: 3
//Número primo: 5
//Número primo: 7
//Total de números primos: 4
ContarNumerosPrimos(10) == 4//true
```

- 1.4 Implemente a função abaixo que conta e calcula a quantidade de vogais dentro de uma string.

```
 CalcularVogais("Luby Software") == 4//true
```

- 1.5 Implemente a função abaixo que aplica uma porcentagem de desconto a um valor e retorna o resultado.
- Lembre-se que as entradas e saídas dos dados são strings que devem ser tratadas.

```
 CalcularValorComDescontoFormatado("R$ 6.800,00", "30%") == "R$ 4.760,00"; //true
```

- 1.6 Implemente a função abaixo que obtém duas string de datas e calcula a diferença de dias entre elas.

```
 CalcularDiferencaData("10122020", "25122020") == 15; //true
```

- 1.7 Implemente a função abaixo que retorna um novo vetor com todos elementos pares do vetor informado.

```
int[] vetor = new int[] { 1,2,3,4,5 };
ObterElementosPares(vetor) == new int { 2, 4 }; //true
```

- 1.8 Implemente a função abaixo que deve buscar um ou mais elementos no vetor que contém o valor ou parte do valor informado na busca.

```
string[] vetor = new string[] {
    "John Doe",
    "Jane Doe",
    "Alice Jones",
    "Bobby Louis",
    "Lisa Romero"
};

BuscarPessoa(vetor, "Doe") == new string[] { "John Doe", "Jane Doe" };//true
BuscarPessoa(vetor, "Alice") == new string[] { "Alice Jones" };//true
BuscarPessoa(vetor, "James") == new string[] { };//true
```

- 1.9 Implemente a função abaixo que obtém uma string com números separados por vírgula e transforma em um array de array de inteiros com no máximo dois elementos.

```
TransformarEmMatriz("1,2,3,4,5,6") == new int[][] { new int[] { 1, 2 }, new int[] { 3, 4 }, new int[] { 5, 6 } }; //true
```

- 1.10 Implemente a função abaixo que compara dois vetores e cria um novo vetor com os elementos faltantes de ambos.

```
// faltam elementos no vetor2
int[] vetor1 = new int[] { 1,2,3,4,5 };
int[] vetor2 = new int[] { 1,2,5 };
ObterElementosFaltantes(vetor1, vetor2) == new int[] { 3, 4 }; //true

// faltam elementos no vetor3
int[] vetor3 = new int[] { 1,4,5 };
int[] vetor4 = new int[] { 1,2,3,4,5 };
ObterElementosFaltantes(vetor3, vetor4) == new int[] { 2, 3 }; //true

// faltam elementos em ambos vetores
int[] vetor5 = new int[] { 1,2,3,4 };
int[] vetor6 = new int[] { 2,3,4,5 };
ObterElementosFaltantes(vetor5, vetor6) == new int[] { 1, 5 }; //true

// não faltam items
int[] vetor7 = new int[] { 1,3,4,5 };
int[] vetor8 = new int[] { 1,3,4,5 };
ObterElementosFaltantes(vetor7, vetor8) == new int[] { }; //true
```

2 SQL

> Resolva as questões utilizando as tabelas abaixo como referência\

```
+----+--------------+
| tabela_pessoa     |
+----+--------------+
| id | nome         |
+----+--------------+
|  1 | John Doe     |
|  2 | Jane Doe     |
|  3 | Alice Jones  |
|  4 | Bobby Louis  |
|  5 | Lisa Romero  |
+----+--------------+
```

```
+----+----------------+-----------+
| tabela_evento                   |
+----+----------------+-----------+
| id | evento         | pessoa_id |
+----+----------------+-----------+
|  1 | Evento A       |  2        |
|  2 | Evento B       |  3        |
|  3 | Evento C       |  2        |
|  4 | Evento D       |  NULL     |
+----+----------------+-----------+
```

- 2.1 Crie uma query para selecionar todas as pessoas 'tabela_pessoa' e os respectivos eventos 'tabela_evento' o qual elas participam.
- 2.2 Crie uma query para selecionar todas as pessoas 'tabela_pessoa' com sobrenome 'Doe'.
- 2.3 Crie uma query para adicionar um novo evento 'tabela_evento' e relacionar à pessoa 'Lisa Romero'.
- 2.4 Crie uma query para atualizar 'Evento D' na 'tabela_evento' e relacionar à pessoa 'Joh Doe'
- 2.5 Crie uma query para remover o 'Evento B' na 'tabela_evento'.
- 2.6 Crie uma query para remover todas as pessoas 'tabela_pessoa' que não possuem eventos 'tabela_evento' relacionados.

> Gerênciamento de tabelas

- 2.7 Crie uma query para alterar a tabela 'tabela_pessoa' e adicionar a coluna 'idade' (int)
- 2.8 Crie uma query para criar uma tabela chamada 'tabela_telefone' que pertence a uma pessoa com seguintes campos:\

```
id: int (PK)
telefone: varchar(200)
pessoa_id: int(FK)
```

- 2.9 Crie uma query para criar uma índice do tipo **único** na coluna telefone da 'tabela_telefone'.
- 2.10 Crie uma query para remover a 'tabela_telefone'.

---

3 Desafio em Orientação a Objetos

Desenvolver programa que rode uma Vending Machine (Máquina de venda de bebidas em lata) utilizando orientação objetos conforme as regras abaixo.

> - Crie uma interface de usuário simples para execução da máquina. (Utilizando Console por exemplo)
> - A máquina deverá possuir um estoque de produtos com valor e quantidade de cada produto. A quantidade de produto no estoque da máquina deve ser alterado conforme realização de vendas dos produtos.
> - A máquina deverá ter opção para visualizar estoque e quantidade disponível.
> - A máquina só pode vender produtos com quantidade em estoque disponível.
> - A máquina deverá contabilizar as vendas e mostrar o valor total das vendas realizadas.
> - Uma venda só poderá ser concluída ao inserir o valor total do produto.
> - A máquina deverá contabilizar e solicitar o valor faltante para finalizar a venda, caso haja valor de troco para deverá informar o valor.
> - A máquina não necessita de lógica de contagem de notas, será apenas necessário informar os valores.
> - Caso necessário crie um documento simples com informações de como executar o programa.