Faculdade IMPACTA	**AVALIAÇÃO FINAL**

ORIENTAÇÕES SOBRE AVALIAÇÃO FINAL

Disciplina: AUTOMAÇÃO DE TESTES DE SOFTWARE	
Cursos: Tecnólogo em Análise e Desenvolvimento de Sistemas Sistemas de Informação	**Semestre**: 1S 2024 **Data de entrega:** 31 de maio de 2024

A avaliação final deste semestre será feita em grupo com no máximo **3 pessoas.**

1) [2.0 pontos] Considere a função abaixo

```python
def verifica_triangulo(a, b, c):
    # Testando se é triângulo
    if (a + b < c) or (a + c < b) or (b + c < a):
        return 'Nao é um triangulo'
    elif (a == b) and (a == c) :
        return 'Equilatero'
    elif (a==b) or (a==c) or (b==c):
        return 'Isósceles'
    else:
        return 'Escaleno'
```

   a) Determine os casos de testes para a função verifica_triangulo(), e escreva um arquivo test_triangulo.py utilizando o pytest.

2) [4.0 pontos] Criar um script utilizando Selenium que acessa uma página que contém letras de músicas, obtenha e imprima na tela a letra da música de sua preferência.

3) [4.0 pontos] Considere a descrição do escopo a seguir:

- Deseja-se especificar um software para controlar um estacionamento.
- Quando um automóvel entra no estacionamento, o frentista deve emitir um ticket com um identificador único, o número do marcador (normalmente um cubo amarelo numerado para facilitar a busca pelo automóvel dentro do estacionamento), data e hora de entrada, que constarão na impressão, juntos com número da placa e o modelo do veículo.
- O ticket, após emitido, é impresso e entregue ao cliente. Quando o cliente retorna para retirar o veículo, ele apresenta o ticket ao frentista, que calcula o valor devido com base no tempo decorrido (R$ 15,00 a primeira hora e R$ 5,00 às demais) e dá baixa no ticket após o pagamento.

Faculdade IMPACTA

AVALIAÇÃO FINAL

Considere as classes a seguir:

```python
from datetime import datetime
class Estacionamento:
    def __init__(self):
        self.tickets_em_aberto = []

    def emitir_ticket(self, ticket):
        self.tickets_em_aberto.append(ticket)

    def registrar_saida(self, ticket):
        ticket.saida = datetime.now()

    def calcular_valor_devido(self, ticket):
        if ticket.entrada is None or ticket.saida is None:
            raise ValueError("Ticket inválido: entrada ou saída não registrada.")

        tempo_decorrido = ticket.saida - ticket.entrada
        horas_decorridas = tempo_decorrido.total_seconds() / 3600  # Converter para horas
        valor_devido = 15 + max(0, horas_decorridas - 1) * 5  # R$ 15 para a primeira hora, R$ 5 para as demais

        return valor_devido
```

```python
from datetime import datetime


class Ticket:
    def __init__(self, placa, modelo, entrada=None, saida=None):
        self.placa = placa
        self.modelo = modelo
        self.entrada = entrada or datetime.now()
        self.saida = saida
```

Considere o arquivo de feature "estacionamento.feature" a seguir:

```
# arquivo: estacionamento.feature

Feature: Controle de Estacionamento

  Scenario: Emitir um ticket para entrada de veículo
    Given um veículo entra no estacionamento
    When o frentista emite um ticket para o veículo
    Then o ticket contém informações corretas sobre a entrada do veículo

  Scenario: Calcular o valor devido para a saída de veículo
    Given um veículo está estacionado há 2 horas
    When o cliente apresenta o ticket de entrada para a saída
    Then o frentista calcula o valor devido corretamente
```

Com os arquivos "ticket.py" e “estacionamento.py” gerar o arquivo test_estacionamento.py. Escreva os testes automatizados utilizando pytest-bdd.