Prova 1

Nome: Adriano Mendes Lima

Matrícula: 508201

Obs: Caso não esteja vendo código completo dentro das caixas de texto, expanda-a para ficar igual ao pdf que mostra como seria a apresentação desejada. Escreva suas respostas no .doc.

1. Cada uma dessas subquestões contém um bloco de código. Trate cada bloco de código de forma independente (ou seja, código em uma pergunta não está relacionado ao código em outra).

	Experiência?	
Idade	Sim	Não
menor de 18	R$12	R$9,5
18 e acima	R$15	R$10,5

a) A tabela à direita mostra como a idade e a experiência de um funcionário afetam seu salário por hora. Suponha que você tenha uma variável booleana **experiencia** e uma variável inteira **idade** que corresponda aos rótulos na tabela. Preencha as condições no código abaixo para calcular a hora de trabalho do funcionário.

```
if experiencia == True:
    if idade < 18:
        hora_trabalho = 12
    else:
        hora_trabalho = 15
else:
    if idade < 18 :
        hora_trabalho = 9.5
    else:
        hora_trabalho = 10.5
```

b) Preencha o espaço em branco para que, quando esse código for executado, o usuário seja solicitado a inserir dois números e, em seguida, a média desses números seja impressa. As taxas de pagamento provavelmente contêm valores decimais.

num1 = input("Digite o valor de sua hora de trabalho: ")
num2 = input("Digite o valor da hora de trabalho do seu colega: ")

print("A média da hora de trabalho é", `(float(num1) + float(num2)) / 2`)

c) Preencha o espaço em branco para chamar elevacao_cidade para obter a elevação (altura acima do nível do mar) de Russas.

```
def elevacao_cidade(cidade: str) -> float:
    """ Retorna a elevação de uma cidade."""
    ... (Resto do código não mostrado)
    return elevacao
```

print("A elevação de Russas é:", `elevacao_cidade("Russas")`)

2. Complete o código abaixo. Lembre-se de colocar casos de teste na descrição.
   a) Complete a função **categoria_ovo** que retorna uma **str** descrevendo a categoria de um ovo dado seu peso **int** em gramas. Aqui está uma tabela que especifica as faixas de peso - se o peso de um ovo está no limite entre duas faixas de categoria, ele é atribuído à categoria menor.

Categoria	Peso
Pequeno	não mais que 50 gramas
Médio	50-57 gramas
Grande	57-64 gramas
Extra Grande	mais que 64 gramas

Resposta:

```
def categoria_ovo(peso: int) -> str:
    """Retorna uma str descrevendo a categoria de um ovo de acordo com o peso"""
    if peso <= 50:
        print('Pequeno')
    elif peso > 50 and peso <= 57:
        print('Médio')
    elif peso > 57 and peso <= 64:
        print('Grande')
    elif peso > 64:
        print('Extra Grande')
    return (f'O ovo pesa: {peso} gramas')

peso = int(input("Digite o peso: "))

print(categoria_ovo(peso))
```

b) Escreva a docstring para a função seguinte.

Resposta:

```
def func(n: int) -> str:
    """ Dado um número 'n', a função declara a variavel 'total' que inicialmente é
        zero, e enquanto 'n' for maior que zero irá somar 'total' ao produto de 'n*n'
        e subtrair 1 de 'n' a cada operação.
    >>> func(5)
    55
    >>> func(10)
    385
    """
    total = 0
    while n > 0:
        total += n * n
        n -= 1
    return total
```

3. Complete a função abaixo de acordo com a docstring.

Resposta:

```
def sequencia_mais_longa(string_busca: str, ch: str) -> int:
    """
    Retorna o comprimento da sequência mais loga de caracteres ch em
sequência na string string_busca. Por exemplo:

    >>> sequencia_mais_longa("aababbbabb", "b")
    3
    >>> sequencia_mais_longa("aababbbabb", "a")
    2
    """
    vazio = []
    num = 0
    for sequencia in string_busca:
        if vazio == "":
            vazio = sequencia

        if sequencia == ch:
            num += 1
        else:
            num = 0

        vazio.append(num)
        return max(vazio)

print(sequencia_mais_longa("aababbbabb", "a"))
```

4. Preencha o código abaixo para completar a implementação da função. Ela substitui todas as ocorrências da substring str_b pela substring str_a em string_orig, usando recursão.
Resposta:

```
def substitui(string_orig: str, str_a: str, str_b: str) -> str:
        #caso base
        if not string_orig:
           return ""

        #caso recursivo
        string_subst = string_orig.replace(str_a, str_b)
        if string_subst == string_orig:
           return string_subst

        return substitui(string_subst, str_a, str_b)

print(substitui("melancia", "melan", "cia"))
```

5. Escreva uma função com protótipo **def somabit(bit1: int, bit2: int, vai_um: int) -> int,int:** que recebe como argumentos três inteiros com valores 0 ou 1 (bit1, bit2 e vai_um) e retorna dois valores inteiros: o vai um e o resultado da soma dos três valores(vai_um, bit1 e bit2). Escreva um programa em python que lê dois números binários, de mesmo tamanho, e calcula um binário que é a soma dos binários lidos do terminal. Utilize a função acima na sua resolução.

Obs: Para saber mais sobre a soma binária, acesse este link. E para retornar mais de uma valor em python basta separar os valores a serem retornado por vírgula. Exemplo: return valor1, valor2. Assim, se chamarmos uma função que retorna mais de um valor, podemos salvar em diversas variáveis. Exemplo: var1, var2 = funcao_exemplo(). Essa função retornaria dois valores onde o primeiro será armazenado em var1 e o segundo em var2.
Resposta:

```
bit1 = int(input("Valor 1: "))
bit2 = int(input("Valor 2: "))
vai_um = int(input("Valor 3: "))

def somabit(bit1: int, bit2: int, vai_um: int) -> int:
  soma = bin(bit1 + bit2 + vai_um)
  return vai_um, soma

print(somabit(bit1, bit2, vai_um))
```