# Exercício – Teste Unitário Automatizado

**Descrição:**

Cada integrante do grupo deve escolher dois problemas da lista abaixo e desenvolver a solução (algoritmo) utilizando apenas uma função. Os algoritmos implementados devem ser testados, de forma automática, utilizando o Jest.

**Observações:**

- Os integrantes do grupo devem realizar a **divisão dos problemas**, para que o mesmo não seja resolvido por dois ou mais do mesmo grupo;
- Os **casos de testes** devem ser **armazenados em um arquivo separado** do teste automático, de preferência no formato JSON;
- Cada algoritmo deve possuir pelo menos **50 casos de teste**.
- **Outros comparadores, além do básico `toBe()`, devem ser utilizados**;
- **Os critérios: particionamento de equivalência, análise do valor limite e error-guessing devem ser utilizados para elaboração de casos de testes;**
- Para complementar os 50 casos de teste para cada problema, **outras ferramentas podem ser utilizadas para geração de dados de teste, como o TA (teste aleatório)** através do mockaroo (https://www.mockaroo.com/).

**Entrega:**

- Os arquivos gerados devem ser colocados em **uma pasta "Trabalhos" na raiz do repositório de cada grupo**;
- Subpastas podem ser criadas para organizar o trabalho de cada integrante;
- **Cada integrante deve enviar realizar o envio dos arquivos relacionados ao seu próprio trabalho para o repositório**.

## Data de entrega: 19/08/2022

## Lista de problemas:

1. Escrever uma função que encontre uma palavra dentro de uma string
   *Exemplo entrada: "hoje é meu dia preferido de aula remota", "dia"*
   *Exemplo saída: 'dia' encontrada uma vez*

2. Escrever uma função que esconda parte dos endereços de e-mail para proteger os dados dos usuários
   *Exemplo entrada: meuemail@gmail.com*
   *Exemplo saída: meu****@gmail.com*

3. Escrever uma função que parametrize uma string
   *Exemplo entrada: "hoje é dia de aula remota"*
   *Exemplo saída: hoje-é-dia-de-aula-remota*

4. Escreva uma função para truncar uma frase dado um determinado número de palavras
   *Exemplo entrada: "hoje é meu dia preferido de aula remota", 5*
   *Exemplo saída: "hoje é meu dia preferido"*

5. Escreve uma função que tenha como entrada um número e insira um traço ("-") entre dois números pares.
   *Exemplo entrada: 2876418*
   *Exemplo saída: 2-876-418*

6. Escreva uma função que encontre o item mais frequente em um array.
   *Exemplo entrada: [7,'b',2,7,'b',6,4,'b',1,9,2];*
   *Exemplo saída: 'b' 3 vezes*

7. Escreva uma função que faça o merge de dois arrays e remova os elementos duplicados.
   *Exemplo entrada: [8, 4, 5, 7], [3, 2, 1, 8]*
   *Exemplo saída: [8, 4, 5, 7, 3, 2, 1]*

8. Escreva uma função que calcule o número de vogais em uma string
   *Exemplo entrada: "praticar exercícios faz bem"*
   *Exemplo saída: 12 vogais*

9. Escreva uma função que some os dígitos do número recebido por parâmetro. Se o resultado desta soma for um número par, a função deve retornar "soma par", caso contrário, se o resultado da soma for um número ímpar, a função deve retornar "soma ímpar".
   *Exemplo entrada: 78923464578*
   *Exemplo saída: "soma ímpar"*

10. Considerando que as letras podem assumir valores de acordo com sua posição no alfabeto (a=1, b=2, c=3, d=4, ....) podemos ter o conceito de palavra balanceada. Uma palavra balanceada é aquela em que a soma dos valores a esquerda é igual a soma dos valores direita. Para palavras

com número ímpar de letras, ignora-se a do meio. Escreva uma função que receba uma string por parâmetro e retorne *true* se ela for balanceada ou *false* se não for.
*Exemplo entrada: "amora"*
*Exemplo saída: false*

11. Escreva uma função que reverta os dígitos de um número inteiro.
    *Exemplo entrada: 123*
    *Exemplo saída: 321*

    *Exemplo entrada: 120*
    *Exemplo saída: 21*

12. Escreva uma função que receba uma string e retorne o tamanho da maior substring que sem caracteres que se repitam.
    *Exemplo entrada: "abcabcbb"*
    *Exemplo saída: 3*

    *Exemplo entrada: "bbbbb"*
    *Exemplo saída: 1*