Módulo Strings

Curso Estruturas de Dados e Algoritmos Expert  
Prof. Dr. Nelio Alves  
<https://devsuperior.com.br>

**Lista de exercícios**Soluções:

<https://github.com/devsuperior/curso-eda/tree/main/strings>

**Problema "cpf"**

Dado o CPF de uma pessoa, o qual pode conter pontos ou traços como separadores, retorne o CPF contendo somente dígitos.

Exemplo 1:

| **Entrada** | **Saída** |
| --- | --- |
| 87409217293 | 87409217293 |

Exemplo 2:

| **Entrada** | **Saída** |
| --- | --- |
| 874092172-93 | 87409217293 |

Exemplo 3:

| **Entrada** | **Saída** |
| --- | --- |
| 874.092.172-93 | 87409217293 |

Assinaturas:

Javascript:

function removeNonDigits(str)

Java:

public static String removeNonDigits(String str)

C#:

public static string RemoveNonDigits(string str)

Python:

def remove\_non\_digits(string)

**Problema "dominio-email"**

Dado um email de uma pessoa, mostrar o nome de usuário e o domínio deste email. O programa deve também informar se o domínio é brasileiro (.br). Supor um email válido.

Exemplo 1:

| **Entrada** | **Saída** |
| --- | --- |
| joao.silva23@yahoo.com.br | Usuario: joao.silva23  Dominio: yahoo.com.br  Brasileiro: sim |

Exemplo 2:

| **Entrada** | **Saída** |
| --- | --- |
| maria123@gmail.com | Usuario: maria123  Dominio: gmail.com  Brasileiro: nao |

Assinaturas:

Javascript:

function extractEmailInformation(email)

Java:

public static EmailInfo extractEmailInformation(String email)

C#:

public static EmailInfo ExtractEmailInformation(string email)

Python:

def extract\_email\_information(email)

**Problema "data1"**

Dada uma data que poderá estar no formado dia/mês/ano, sendo que o dia e mês menor que 10 pode ou não ter zero à esquerda, mostrar o dia, mês e ano separadamente, na forma numérica. Supor uma data válida.

Exemplo:

| **Entrada** | **Saída** |
| --- | --- |
| 21/07/2010 | Dia: 21  Mês: 7  Ano: 2010 |

Assinaturas:

Javascript:

function extractDateData(date)

Java:

public static DateInfo extractDateData(String date)

C#:

public static DateInfo ExtractDateData(string date)

Python:

def extract\_date\_data(date)

**Problema "data2"**

Dados um dia, mês e ano, produzir um string no formato dd/mm/aaaa, preenchendo com zeros à esquerda o dia e mês, se necessário.

Exemplo:

| **Entrada** | **Saída** |
| --- | --- |
| {  "day": 21,  "month": 7,  "year": 2010  } | 21/07/2010 |

Assinaturas:

Javascript:

function formatDate(day, month, year)

Java:

public static String formatDate(int day, int month, int year)

C#:

public static string FormatDate(int day, int month, int year)

Python:

def format\_date(day, month, year)

**Problema "senha"**

Um sistema exige que, no cadastro, a senha do usuário tenha pelo menos 8 caracteres no total, pelo menos uma letra (maiúscula ou minúscula), pelo menos um dígito e pelo menos um caractere especial: @, #, &. Fazer um programa para dizer se uma dada senha é válida ou inválida.

Exemplo 1:

| **Entrada** | **Saída** |
| --- | --- |
| amerca1@ | VALIDA |

Exemplo 2:

| **Entrada** | **Saída** |
| --- | --- |
| amrca154682 | INVALIDA |

Assinaturas:

Javascript:

function validatePassword(str)

Java:

public static boolean validatePassword(String str)

C#:

public static bool ValidatePassword(string str)

Python:

def validate\_password(string)

**Problema "valid-anagram"** *(ref: Leetcode valid-anagram)*

*Empresas: Apple, Google, Amazon, Microsoft, Adobe, Spotify, etc.*

Dadas duas palavras s e t, retornar true se t é um anagrama de s, ou false caso contrário.

Um anagrama é um texto formado pelo rearanjo das letras em um texto diferente, tipicamente utilizando todas letras originais exatamente uma vez.

Limitações:

* 1 <= s.length, t.length <= 5 \* 104
* s e t são letras minúsculas da Língua Inglesa.

Exemplo 1:

| **Entrada** | **Saída** |
| --- | --- |
| anagram  nagaram | true |

Exemplo 2:

| **Entrada** | **Saída** |
| --- | --- |
| rat  car | false |

Assinaturas:

Javascript:

function isAnagram(s, t)

Java:

public static boolean isAnagram(String s, String t)

C#:

public static bool IsAnagram(string s, string t)

Python:

def is\_anagram(s, t)

**Problema "prefixo-comum"** *(ref: Leetcode* longest-common-prefix*)*

*Empresas: Apple, Google, Amazon, Microsoft, Adobe, Accenture, etc.*

Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string "".

Constraints:

* 1 <= strs.length <= 200
* 0 <= strs[i].length <= 200
* strs[i] consists of only lowercase English letters.

Exemplo 1:

| **Entrada** | **Saída** |
| --- | --- |
| ["flowers","flow","flight"] | "fl" |

Exemplo 2:

| **Entrada** | **Saída** |
| --- | --- |
| ["dog","racecar","car"] | "" |

Assinaturas:

Javascript:

function longestCommonPrefix(v)

Java:

public static String longestCommonPrefix(String[] v)

C#:

public static string LongestCommonPrefix(string[] v)

Python:

def longest\_common\_prefix(v)

**Problema "transacoes"** *(ref: Leetcode* invalid-transactions*)*

*Empresas: Apple, Google, Amazon, Microsoft, Adobe, Accenture, etc.*

A transaction is possibly invalid if:

* the amount exceeds $1000, or;
* if it occurs within (and including) 60 minutes of another transaction with the same name in a different city.

You are given an array of strings transaction where transactions[i] consists of comma-separated values representing the name, time (in minutes), amount, and city of the transaction. Return a list of transactions that are possibly invalid. You may return the answer in any order.

Constraints:

* transactions.length <= 1000
* Each transactions[i] takes the form "{name},{time},{amount},{city}"
* Each {name} and {city} consist of lowercase English letters, and have lengths between 1 and 10.
* Each {time} consist of digits, and represent an integer between 0 and 1000.
* Each {amount} consist of digits, and represent an integer between 0 and 2000.

Exemplo 1:

| **Entrada** |
| --- |
| ["alice,20,800,mtv","alice,50,100,beijing"] |
| **Saída** |
| ["alice,20,800,mtv","alice,50,100,beijing"] |

Exemplo 2:

| **Entrada** |
| --- |
| ["alice,20,800,mtv","alice,50,1200,mtv"] |
| **Saída** |
| ["alice,50,1200,mtv"] |

Exemplo 3:

| **Entrada** |
| --- |
| ["alice,20,800,mtv","bob,50,1200,mtv"] |
| **Saída** |
| ["bob,50,1200,mtv"] |

Assinaturas:

Javascript:

function invalidTransactions(transactions)

Java:

public static List<String> invalidTransactions(String[] transactions)

C#:

public static List<string> InvalidTransactions(string[] transactions)

Python:

def invalid\_transactions(transactions)