# **DESAFIO DE PROGRAMAÇÃO - ACADEMIA CAPGEMINI**

Olá! Seja bem-vindo (a) à terceira etapa do processo de seleção para a Academia Capgemini 2022. O objetivo dessa etapa é testar os seus conhecimentos em lógica de programação. Para isso, preparamos três questões com diferentes níveis de dificuldade. *A implementação das questões pode ser feita em qualquer linguagem, porém a utilização de Java será um diferencial.*

## **# Questão 01**

Escreva um algoritmo que mostre na tela uma escada de tamanho **n** utilizando o caractere **\*** e espaços. A base e altura da escada devem ser iguais ao valor de **n.** A última linha não deve conter nenhum espaço.

**Exemplo:**

**Entrada:**

|  |
| --- |
| n = 6 |

**Saída:**

|  |
| --- |
| **\*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*** \*\*\*\*\*\* |

Resposta:

package desafio;

import java.util.Scanner;

public class Exercicio1 {

public static void main ( String [] args) {

@SuppressWarnings("resource")

Scanner s = new Scanner(System.in);

int n = s.nextInt();

for (int i = 1; i < n; i++) {

for(int j = 1; j<=n-1; j++)

System.out.print("");

for (int j= 1;j<=2\*i-1; j++)

System.out.print("\*");

System.out.println();

}

}

}

## **}# Questão 02**

Débora se inscreveu em uma rede social para se manter em contato com seus amigos. A página de cadastro exigia o preenchimento dos campos de nome e senha, porém a senha precisa ser forte. O site considera uma senha forte quando ela satisfaz os seguintes critérios:

* Possui no mínimo 6 caracteres.
* Contém no mínimo 1 digito.
* Contém no mínimo 1 letra em minúsculo.
* Contém no mínimo 1 letra em maiúsculo.
* Contém no mínimo 1 caractere especial. Os caracteres especiais são: **!@#$%^&\*()-+**

Débora digitou uma string aleatória no campo de senha, porém ela não tem certeza se é uma senha forte. Para ajudar Débora, construa um algoritmo que informe qual é o número mínimo de caracteres que devem ser adicionados para uma string qualquer ser considerada segura.

**Exemplo:**

**Entrada:**

|  |
| --- |
| Ya3 |

**Saída:**

|  |
| --- |
| 3 |

**Explicação:**

Ela pode tornar a senha segura adicionando 3 caracteres, por exemplo, &ab, transformando a senha em Ya3&ab. 2 caracteres não são suficientes visto que a senha precisa ter um tamanho mínimo de 6 caracteres.

Resposta:

package desafio2;

import java.util.Arrays;

import java.util.HashSet;

import java.util.Set;

public class Exercicio2 {

static class Exercicio {

public static void printStrongNess(String input)

{

int n = input.length();

boolean hasLower = false, hasUpper = false,

hasDigit = false, specialChar = false;

Set<Character> set = new HashSet<Character>(

Arrays.asList('!', '@', '#', '$', '%', '^', '&',

'\*', '(', ')', '-', '+'));

for (char i : input.toCharArray())

{

if (Character.isLowerCase(i))

hasLower = true;

if (Character.isUpperCase(i))

hasUpper = true;

if (Character.isDigit(i))

hasDigit = true;

if (set.contains(i))

specialChar = true;

}

System.out.print("Strength of password:- ");

if (hasDigit && hasLower && hasUpper && specialChar

&& (n >= 6))

System.out.print(" Strong");

else if ((hasLower || hasUpper || specialChar)

&& (n >= 6))

System.out.print(" Moderate");

else

System.out.print(" Weak");

}

public static void main(String[] args)

{

String input = "kkkk12";

printStrongNess(input);

}

}

}

**# Questão 03**

Duas palavras podem ser consideradas anagramas de si mesmas se as letras de uma palavra podem ser realocadas para formar a outra palavra. Dada uma string qualquer, desenvolva um algoritmo que encontre o número de pares de substrings que são anagramas.

**Exemplo:**

**Exemplo 1)**

**Entrada**:

|  |
| --- |
| ovo |

**Saída:**

|  |
| --- |
| 2 |

**Explicação:**

A lista de todos os anagramas pares são: [o, o], [ov, vo] que estão nas posições [[0], [2]], [[0, 1], [1, 2]] respectivamente.

**Exemplo 2)**

**Entrada:**

|  |
| --- |
| ifailuhkqq |

**Saída:**

|  |
| --- |
| 3 |

**Explicação:**

Resposta:

package desafio3;

import java.util.Arrays;

public class Exercicio3 {

static class GFG {

static boolean areAnagram(char[] str1, char[] str2)

{

int n1 = str1.length;

int n2 = str2.length;

if (n1 != n2)

return false;

Arrays.sort(str1);

Arrays.sort(str2);

for (int i = 0; i < n1; i++)

if (str1[i] != str2[i])

return false;

return true;

}

/\* Driver Code\*/

public static void main(String args[])

{

char str1[] = { 't', 'e', 's', 't' };

char str2[] = { 't', 't', 'e', 'w' };

if (areAnagram(str1, str2))

System.out.println("The two strings are"

+ " anagram of each other");

else

System.out.println("The two strings are not"

+ " anagram of each other");

}

}

}

# O que será avaliado

* Documentação
* Estrutura do código
* Atendimento aos requisitos
* Testes unitários

## **# Envio das questões**

As soluções para as questões devem ser hospedadas no GitHub e o link do repositório deve ser postado na sua área do candidato a partir do dia 14/02/2022**.** Para entrar na sua área do candidato acesse: <https://capgemini.proway.com.br/inscricao/login.php>. O link do repositório deve ser postado no campo **“Github para o desafio de programação”**. O link deve ser similar a este: <https://github.com/nome-de-usuario/repositorio>. Lembrando que a data final para postagem do desafio será no dia **20/02/2022.** Quanto antes você fizer, maiores as chances de ser selecionado (a) para a próxima etapa. 🚀

O repositório deve conter um arquivo README.md com as instruções de como rodar a aplicação e as tecnologias utilizadas.