# **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:

import java.util.Scanner;

public class Main {

public static void main ( String [] args) {

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:

import java.util.regex.Pattern;

import java.util.regex.Matcher;

class GFG {

static int countCharacters(

String password)

{

int count = 0;

Pattern digit = Pattern.compile("(\\d)");

Pattern upper = Pattern.compile("([A-Z])");

Pattern lower = Pattern.compile("([a-z])");

Pattern spChar = Pattern.compile("([\\W)](file:///\\W))");

Matcher Digit = digit.matcher(password);

Matcher Upper = upper.matcher(password);

Matcher Lower = lower.matcher(password);

Matcher Special = spChar.matcher(password);

if (!Digit.find()) {

count++;

}

if (!Upper.find()) {

count++;

}

if (!Lower.find()) {

count++;

}

if (!Special.find()) {

count++;

}

if ((count + password.length()) < 6) {

count = count + 6

- (count + password.length());

}

return count;

}

public static void main(String args[])

{

String password1 = "Karensforkarens";

System.out.println(

countCharacters(password1));

String password2 = "Karens1";

System.out.println(

countCharacters(password2));

}

}

**# 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:**

public class KAR

{

static final int MAX = 256;

static boolean compare(char arr1[], char arr2[])

{

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

if (arr1[i] != arr2[i])

return false;

return true;

}

static void search(String pat, String txt)

{

int M = pat.length();

int N = txt.length();

// countP[]: Store count of all

char[] countP = new char[MAX];

char[] countTW = new char[MAX];

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

{

(countP[pat.charAt(i)])++;

(countTW[txt.charAt(i)])++;

}

for (int i = M; i < N; i++)

{

if (compare(countP, countTW))

System.out.println("Found at Index " +

(i - M));

(countTW[txt.charAt(i)])++;

countTW[txt.charAt(i-M)]--;

}

if (compare(countP, countTW))

System.out.println("Found at Index " +

(N - M));

}

public static void main(String args[])

{

String txt = "BACDGABCDA";

String pat = "ABCD";

search(pat, txt);

}

}

# 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.