# **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:**

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

**Resolução**

Algoritmo "questao01"

// 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.

Var

// Seção de Declarações das variáveis

v\_retorno : caractere

n : inteiro

i, j, k: inteiro

Inicio

// Seção de Comandos, procedimento, funções, operadores, etc...

escreva ( "Digite um número inteiro para montar uma escada de asterisco: ")

leia (n) //Recebe o valor informado

//Utiliza o valor informado como referência para montar a escada

para i := 1 ate n faca

v\_retorno <- ""

//Executa a rotina para montar a escada

para j := n ate 1 passo -1 faca

se j <= i entao

v\_retorno <- v\_retorno + "\*"

senao

v\_retorno <- v\_retorno + " "

fimse

fimpara

//Mostra o resultado

escreval (v\_retorno)

fimpara

Fimalgoritmo

## **# 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.

**Resolução**

Algoritmo "questao02"

// 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

Var

// Seção de Declarações das variáveis

v\_senha, v\_parcial : caractere

v\_tamanho, v\_asc, v\_retorno : inteiro

v\_mai, v\_minus, v\_num, v\_esp, v\_total, i, v\_qtd\_min, v\_restricao: inteiro

Inicio

// Seção de Comandos, procedimento, funções, operadores, etc...

escreva ( "Digite uma senha: ")

leia (v\_senha)

//Limpando as variáveis

v\_retorno <- 0

v\_mai <- 0

v\_minus <- 0

v\_num <- 0

v\_esp <- 0

v\_total <- 0

v\_qtd\_min <- 6 //Coloca quantidade mínima de caracteres necessários

v\_restricao <- 4 //Coloca quantidade mínima de restrições necessárias

//Contagem do comprimento da senha

v\_tamanho <- compr (v\_senha)

se v\_tamanho < v\_qtd\_min entao

v\_retorno <- v\_qtd\_min - v\_tamanho

escreval ("A senha não possui a quantidade mínima de caracteres que é ", v\_qtd\_min)

fimse

para i := 1 ate v\_tamanho faca

//Retorna o valor da variável v\_senha a partir de i com 1 caractere

v\_parcial <- copia (v\_senha, i, 1))

//Testa as condições pré-definidas no enunciado.Se tiver uma das condições

//então a variável de cada condição vale 1

escolha v\_parcial

//Verifica se possui letra maiúscula

caso "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"

v\_mai <- 1

//Verifica se possui letra minúscula

caso "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"

v\_minus <- 1

//Verifica se possui número

caso "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"

v\_num <- 1

//Verifica se possui caractere especial

caso "!", "@", "#", "$", "%", "&", "\*", "(", ")", "-", "+", "^"

v\_esp <- 1

fimescolha

fimpara

//Se as variáveis condicionais não estiverem suprindo as condições

//então exibe mensagem na tela indicando o que falta na senha e

//armazena o valor para saber quantos caracteres faltam no total

se v\_mai = 0 entao

escreval ("Não há letra maiúscula na senha")

v\_total <- v\_total + 1

fimse

se v\_minus = 0 entao

escreval ("Não há letra minúscula na senha")

v\_total <- v\_total + 1

fimse

se v\_num = 0 entao

escreval ("Não há número na senha")

v\_total <- v\_total + 1

fimse

se v\_esp = 0 entao

escreval ("Não há caractere especial na senha")

v\_total <- v\_total + 1

fimse

//Se a quantidade de restrições for maior que a quantidade de caracteres faltantes, calcula a quantidade que falta

se v\_retorno < v\_total entao

v\_retorno <- v\_total - v\_retorno

fimse

//Exibe o resultado da verificação

se v\_retorno > 0 entao

escreva ("Falta(m) " ,v\_retorno, " caractere(s) para a senha ser segura")

senao

escreva ("A senha é segura")

fimse

Fimalgoritmo

**# 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:**

|  |
| --- |
| 3 |

**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:**

A lista de todos os anagramas pares são: [i, i], [q, q] e [ifa, fai] que estão nas posições [[0, 3]], [[8, 9]] e [[0, 1, 2], [1, 2, 3]].

**Resolução**

Algoritmo "questao03"

//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.

Var

// Seção de Declarações das variáveis

v\_palavra, v\_parcial, v\_anagrama: caractere

v\_tamanho, i, j, k, l, m, x: inteiro

j\_aux, k\_aux, a\_total, b\_total, v\_existe: inteiro

a\_aux, b\_aux : caractere

Inicio

// Seção de Comandos, procedimento, funções, operadores, etc...

escreval ("Digite uma palavra : ", v\_palavra)

//Recebe a palavra para verificação da quantidade de anagramas

leia (v\_palavra)

x <- 0

//Verifica o tamanho da palavra

v\_tamanho <- compr (v\_palavra)

//Laço de repetição para controlar a quantidade de intervalo de caracteres

para i := 1 ate v\_tamanho faca

j\_aux <- 1

//Percorre a palavra dividindo em segmentos

para j := i ate v\_tamanho faca

v\_parcial <- copia (v\_palavra; j\_aux; i)

k\_aux <- j\_aux

//Percorre a palavra dividindo em segmentos para comparação com o segmento

//selecionado

para k := j ate v\_tamanho faca

v\_anagrama <- copia (v\_palavra; k\_aux; i)

//Se os segmentos estiverem em posições diferentes, verifica se é anagrama

se (k\_aux <> j\_aux) entao

//Se o i for igual a 1 irá comparar as letras

se i = 1 entao

//Se os segmentos forem iguais, é um anagrama

se v\_parcial = v\_anagrama entao

x <- x + 1

fimse

//Se o i for maior que 1 irá comparar os segmentos

senao

//Se os segmentos não forem iguais faz a verificação de anagrama

se v\_parcial <> v\_anagrama entao

a\_total <- 0

b\_total <- 0

v\_existe <- 0

//Navega de caractere em caractere para comparar com segmento do anagrama

para l := 1 ate compr (v\_parcial) faca

a\_aux <- copia (v\_parcial; l; 1)

//Busca e soma o valor da tabela ascii para comparação

a\_total <- a\_total + asc (a\_aux)

//Percorre o segmento do anagrama

para m := 1 ate compr (v\_anagrama) faca

b\_aux <- copia (v\_anagrama; m; 1)

//Na primeira vez, busca e soma o valor da tabela ascii para comparação

se l = 1 entao

b\_total <- b\_total + asc (b\_aux)

fimse

//Se existir caracteres iguais, associa 1 para o verificador

se a\_aux = b\_aux entao

v\_existe <- v\_existe + 1

fimse

fimpara

fimpara

//Compara o valor recebido com o tamanho do segmento

se v\_existe = compr (v\_parcial) entao

//Verifica se a soma dos dois segmentos são iguais

se a\_total = b\_total entao

//Se forem iguais, incrementa 1 na quantidade

x <- x + 1

fimse

fimse

fimse

fimse

fimse

k\_aux <- k\_aux + 1

fimpara

j\_aux <- j\_aux + 1

fimpara

fimpara

//Mostra o resultado da verificação

escreval ("O número de pares de anagramas é : ", x)

Fimalgoritmo

## **# 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.