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
```

ACADEMIA Capgemini

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]].

ACADEMIA Capgemini

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
    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
```

ACADEMIA Capgemini

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.