EJERCICIO 1:

Isograma

Un isograma es una palabra que no tiene letras repetidas. Consideraciones Adicionales:

Un string vacio es un isograma.

La función tiene que ser case insensitive e ignorar acentos.

Si el string tiene mas de una palabra retornar false.

Se tiene que hacer clean up del string antes de comparar.

SOLUCION JAVA 25:

public static boolean isograma(String palabra){

List lista = new ArrayList();

for(char letra:palabra.toCharArray()){

if (lista.contains(letra)){

return false;

}else if(lista.contains(" ")){

return false;

}else{

lista.add(letra);

}

}

return true;

};

IO.println(isograma("caro"));

EJERCICIO 1 C++:

// Se necesita para std::find

#include <algorithm>

#include <iostream>

#include <list>

#include <string>

bool isograma(std::string palabra) {

std::list<char> letras\_vistas;

for (char letra : palabra) {

if (letra == ' ') {

return false;

}

auto it = std::find(letras\_vistas.begin(), letras\_vistas.end(), letra);

if (it != letras\_vistas.end()) {

return false;

} else {

letras\_vistas.push\_back(letra);

}

}

return true;

}

int main() {

std::cout << std::boolalpha;

std::cout << "La palabra 'gato' es un isograma? " << isograma("gato") << std::endl;

std::cout << "La palabra ' ' es un isograma? " << isograma(" ") << std::endl;

std::cout << "La palabra 'perro' es un isograma? " << isograma("perro") << std::endl;

}

EJERCICIO 2:

/\*

\* Escribe un programa que muestre por consola (con un print) los

\* números de 1 a 100 (ambos incluidos y con un salto de línea entre

\* cada impresión), sustituyendo los siguientes:

\* - Múltiplos de 3 por la palabra "fizz".

\* - Múltiplos de 5 por la palabra "buzz".

\* - Múltiplos de 3 y de 5 a la vez por la palabra "fizzbuzz".

\*/

SOLUCION LENGUAJE NATURAL:

Recorrido de un rango(1,100)🡪 vamos validar:

Condiciones:

Si numero /3 = Fizz, Si numero/5 = Buzz, Si numero/3 y numero/5 = FizzBuzz

SOLUCION PYTHON:

# /\*

# \* Escribe un programa que muestre por consola (con un print) los

# \* números de 1 a 100 (ambos incluidos y con un salto de línea entre

# \* cada impresión), sustituyendo los siguientes:

# \* - Múltiplos de 3 por la palabra "fizz".

# \* - Múltiplos de 5 por la palabra "buzz".

# \* - Múltiplos de 3 y de 5 a la vez por la palabra "fizzbuzz".

# \*/

def fizzbuzz():

rango= range(1,101)

for n in rango:

if ( n%3==0 ) & ( n%5==0 ):

print(f'{n}:FizzBuzz')

elif n%5 == 0:

print(f'{n}:Buzz')

elif n%3==0 :

print(f'{n}:Fizz')

else:

print(n)

fizzbuzz()

SOLUCION EN JAVASCRIPT y TYPESCRIPT:

for(let i=1; i < 101; i++){

if ((i%3==0) && (i%5==0)){

console.log(i+": FizzBuzz")

}else if(i%3==0){

console.log(i+":Fizz")

}else if(i%5==0){

console.log(i+": Buzz")

}else{

console.log(i+"")

}

}

SOLUCION JAVA 25+:

for(int i=1; i<101; i++){

if((i%3==0)&&(i%5==0)){

System.out.println(i+": FizzBuzz");

}else if(i %3 ==0 ){

System.out.println(i+": Fizz");

}else if (i %5==0){

System.out.println(i+": Buzz");

}else{

System.out.println(i+"");

}

}

SOLUCION C ++:

#include <iostream>

#include <string>

int main() {

for (int i = 1; i <= 100; ++i) {

// La condición para FizzBuzz debe ser la primera, ya que es la más específica

if (i % 3 == 0 && i % 5 == 0) {

std::cout << i << ": FizzBuzz\n";

} else if (i % 3 == 0) {

std::cout << i << ": Fizz\n";

} else if (i % 5 == 0) {

std::cout << i << ": Buzz\n";

} else {

std::cout << i << "\n";

}

}

}

EJERCICIO 3:

/\*

\* Escribe una función que reciba dos palabras (String) y retorne

\* verdadero o falso (Bool) según sean o no anagramas.

\* - Un Anagrama consiste en formar una palabra reordenando TODAS

\* las letras de otra palabra inicial.

\* - NO hace falta comprobar que ambas palabras existan.

\* - Dos palabras exactamente iguales no son anagrama.

\*/

SOLUCION LENGUAJE NATURAL:

FUNCION QUE RECIBA 2 PALABRAS:

VERIFICAR SI SON IGUALES RETORNAR FALSE

VERIFICAR QUE SE REPITAN LA MISMA CANTIDAD DE LETRAS EN AMBAS PALABRAS RETURNAR TRUE

SI NO FALSE

SOLUCION PYTHON:

from collections import Counter

def anagrama(p1,p2):

if p1.lower() == p2.lower():

return False

conteo\_palabra1=Counter(p1.lower())

conteo\_palabra2=Counter(p2.lower())

for letra,repeticion in conteo\_palabra1.items():

try:

if conteo\_palabra2[letra]== repeticion:

continue

else:

return False

except Exception as e:

return False

return True

print(anagrama("Sol","Luna"))

SOLUCION JAVASCRIPT:

function anagrama(P1, P2){

p1=P1.toLowerCase();

p2=P2.toLowerCase();

if (p1 == p2){

return false;

}

const ordp1= p1.split("").sort().join("");

const ordp2= p2.split("").sort().join("");

if (ordp1 == ordp2){

return true;

}else{

return false;

}

}

console.log(anagrama("Caza","Zaca"))

SOLUCION TYPESCRIPT:

function anagrama(P1:String, P2:String){

  let p1:String=P1.toLowerCase();

  let p2:String=P2.toLowerCase();

  if (p1 == p2){

    return false;

  }

  const ordp1= p1.split("").sort().join("");

  const ordp2= p2.split("").sort().join("");

  if (ordp1 == ordp2){

    return true;

  }else{

    return false;

  }

}

console.log(anagrama("Amor","Roma"))

SOLUCION JAVA 25+:

record Main(){

public boolean anagrama(String p1, String p2){

p1=p1.toLowerCase();

p2=p2.toLowerCase();

if(p1 == p2){

return false;

}

char[] Arp1= p1.toCharArray();

char[] Arp2= p2.toCharArray();

Arrays.sort(Arp1);

Arrays.sort(Arp2);

if(Arrays.equals(Arp1,Arp2)==true){

return true;

}else{

return false;

}

}

}

Main main= new Main();

System.out.println(main.anagrama("Amor","Roma"));

SOLUCION C++:

#include <iostream>

#include <string>

#include <algorithm> // Para std::sort

#include <vector> // Para std::vector

bool anagrama(std::string p1, std::string p2) {

//tolowerCase

std::transform(p1.begin(), p1.end(), p1.begin(),

[](unsigned char c){ return std::tolower(c); });

std::transform(p2.begin(), p2.end(), p2.begin(),

[](unsigned char c){ return std::tolower(c); });

if (p1 == p2) {

return false;

}

// 3. Ordenar los caracteres

std::sort(p1.begin(), p1.end());

std::sort(p2.begin(), p2.end());

// 4. Comparar las cadenas ordenadas

return p1 == p2;

}

int main() {

// 5. Llamada a la función e impresión del resultado

// Usamos std::cout para imprimir y std::boolalpha para mostrar "true" o "false"

// en lugar de 1 o 0.

std::cout << std::boolalpha << anagrama("Amor", "Roma") << std::endl;

std::cout << std::boolalpha << anagrama("Hola", "Olah") << std::endl;

return 0;

}

EJERCICIO 4: SIGUIENTE DIRECTO->

/\*

\* Escribe un programa que imprima los 50 primeros números de la sucesión

\* de Fibonacci empezando en 0.

\* - La serie Fibonacci se compone por una sucesión de números en

\* la que el siguiente siempre es la suma de los dos anteriores.

\* 0, 1, 1, 2, 3, 5, 8, 13...

\*/