

*Techchallenge*  
2022



# RETO 1 PROGRAMACIÓN



Los ejercicios pueden resolverse en cualquiera de los siguientes lenguajes:



C



C++



C#



JavaScript



PHP



Python3



R

# RESULTADOS

Los resultados serán entregados en formulario de forms subiendo los datos del participante y el código.

<https://forms.gle/C7S5pUKtYnEQF5xeA>



# EJERCICIO 1 EDAD PERMITIDA

---

Pide al usuario que ingrese su edad y mostrarás un mensaje correspondiente si esta coincide con las siguientes condiciones:

Más de 30 años: Nunca es tarde para aprender ¿Qué curso tomaremos?  
Entre 29 y 18 años: Es un momento excelente para impulsar tu carrera.  
Menos de 18 años: ¿Sabes hacia dónde dirigir tu futuro? Seguro puedo ayudarte.

# EJERCICIO 2 VALIDACIÓN DE CÓDIGOS POSTALES

---

Un código postal P válido debe cumplir los dos requisitos siguientes:

- P debe ser un número entre el rango 100000 y 999999.
- P no debe contener más de un par de dígitos repetitivos alternos.

Los dígitos repetitivos alternos son dígitos que se repiten inmediatamente después del dígito siguiente. En otras palabras, un par de dígitos repetitivos alternos está formado por dos dígitos iguales que tienen solo un dígito entre ellos.

Por ejemplo:

121426 # Aquí, 1 es un dígito repetitivo alterno.

523563 # Aquí, NO dígito es un dígito repetitivo alterno.

552523 # Aquí, tanto el 2 como el 5 son dígitos repetitivos alternos.

# EJERCICIO 2 VALIDACIÓN DE CÓDIGOS POSTALES

Su tarea es proporcionar dos expresiones regulares `regex_integer_in_range` y `regex_alternating_repetitive_digit_pair`. Dónde:

`regex_integer_in_range` debe coincidir solo con el rango de enteros desde 10000 hasta 999999.

`regex_alternating_repetitive_digit_pair` debe encontrar pares de dígitos repetitivos alternos en una cadena determinada.

La plantilla de código proporcionada utilizará ambas expresiones regulares para verificar si la cadena de entrada P es un código postal válido mediante la siguiente expresión:

`(bool(re.match(regex_integer_in_range, P)) and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)`

# EJERCICIO 2 VALIDACIÓN DE CÓDIGOS POSTALES

## Formato de salida

La salida solo debe ser un string de **falso** o **verdadero**

## Entrada de muestra

```
110000
```

## Salida de muestra

```
False
```

## Explicación 0

110000 : (0, 0) y (0, 0) son dos pares de dígitos alternos. Por lo tanto, es un código postal no válido.

## Nota:

Se otorgará una puntuación de por usar condiciones 'if' en su código.

Tienes que pasar todos los casos de prueba para obtener una puntuación positiva.

# EJERCICIO 3 ORDENA LAS PALABRAS

Se te dan  $n$  palabras. Algunas palabras pueden repetirse. Para cada palabra, genere su número de ocurrencias. El orden de salida debe corresponder con el orden de entrada de aparición de la palabra. Consulte la muestra de entrada/salida para obtener aclaraciones.

Nota: Cada línea de entrada termina con un carácter "\n".

Restricciones:

La suma de las longitudes de todas las palabras no excede  $10^6$

Todas las palabras están compuestas únicamente por letras minúsculas en español.

# EJERCICIO 3 ORDENA LAS PALABRAS

---

## Formato de entrada

La primera línea contiene el entero, n .

Las siguientes líneas contienen cada una una palabra.

## Formato de salida

2 líneas de salida.

En la primera línea, muestra el número de palabras distintas de la entrada.

En la segunda línea, muestra el número de apariciones de cada palabra distinta según su aparición en la entrada.

# EJERCICIO 3 ORDENA LAS PALABRAS

## Explicación

### Sample Input

```
4
bcdef
abcdefg
bcde
bcdef
```

### Sample Output

```
3
2 1 1
```

Hay palabras distintas. Aquí, "bcdef" aparece dos veces en la entrada en la primera y última posición. Las otras palabras aparecen una vez cada una. El orden de las primeras apariciones es "bcdef", "abcdefg" y "bcde" que corresponde a la salida.

# iSURFTE!

```
public static void main(String[] args)
{
    Vert[] data = new Vert[10];
    data[i] = new Vert();
    data[i].X = START;
    System.out.print(Float.toString(data/Y));
}
```