Desafío - Arreglos

- Para realizar este desafío debes haber estudiado previamente todo el material disponible correspondiente a la unidad.
- Una vez terminado el desafío, comprime la carpeta que contiene el desarrollo de los requerimientos solicitados y sube el `.zip` en el LMS.
- Puntaje total: **10 puntos**
- Desarrollo desafío:
    - El desafío se debe desarrollar de manera Individual.

Instrucciones

A continuación se detallan variados desafíos a desarrollar.

Para su correcta evaluación, los programas deben ser almacenados en un archivo comprimido `.zip` de la siguiente manera:

```
desafios.zip
├── filtro_procesos.rb
├── promedio2.rb
├── smartwatch1.rb
└── visitas.rb
```

Imagen 1.
Fuente: Desafío Latam.

Visitas.rb

Crear el programa `visitas.rb` que dada la información de visitas diarias a un sitio web pueda entregar cierta información.

```
visitas = [1000, 800, 250, 300, 500, 2500]
```

Imagen 2.
Fuente: Desafío Latam.

Se pide:

- Crear un método llamado promedio que devuelva la cantidad promedio de visitas en el arreglo.

Tips:

- La Corrección del ejercicio funciona llamando al método promedio, por lo que el método tiene que existir y el valor ser el promedio de cualquier arreglo entregado.
- Puedes probar el programa llamando al método y mostrando el resultado, pero no es necesario que el programa entregue resultado alguno, la revisión se hace llamando al método.

promedio2.rb

Crear el programa `promedio2.rb` con el método `compara_arrays` que reciba 2 arreglos y calcule el promedio de ambos, devolviendo el mayor de los promedios.

Uso:

```
ruby promedio2.rb
```

Imagen 3.
Fuente: Desafío Latam.

Tips:

- Ocupar los métodos de array y string que estudiamos.
- Puedes ocupar el método para calcular el promedio del ejercicio anterior, pero debes agregarlo al archivo nuevo.
- Debes respetar el nombre del método.
- Puedes probar el programa llamando al método y mostrando el resultado, pero no es necesario que el programa entregue resultado alguno, la revisión se hace llamando al método.

Smartwatch1.rb

Un smartwatch muy inteligente cuenta la cantidad de pasos diarios que da una persona, pero en algunos casos genera información errónea. Se pide crear un método llamado `clear_steps` que reciba un arreglo y descarte todos los valores que no sean números o sean menores a 200 o mayor a 100000. Los valores deben quedar como enteros (Integers). El método debe retornar el arreglo filtrado. El programa debe llamarse `smartwatch1.rb`.

Uso:

```
ruby smartwatch1.rb
```

> El programa no genera output

Imagen 4.
Fuente: Desafío Latam.

Probar el programa con el siguiente arreglo.

```
pasos = ['100', '21', '231as', '2031', '1052000', '213b', 'b123']
```

Tips:

- En la corrección el array entregado al método puede ser distinto al del ejemplo.
- Puedes probar el programa llamando al método y mostrando el resultado pero no es necesario que el programa entregue resultado alguno, la revisión se hace llamando al método.

Filtro_procesos.rb

Se necesita crear un programa llamado `filtro_procesos.rb` que lea un archivo que tiene datos por línea. Estos datos representan la cantidad de milisegundos que demoran en terminar algunos procesos del sistema operativo.

Ejemplo de archivo

procesos.data

```
121
141
159
131
199
121
299
312
412
024
213
```

Imagen 5.
Fuente: Desafío Latam.

Se necesita un programa que pueda leer un archivo de las mismas características y generar un archivo llamado `procesos_filtrados.data` donde todos los valores sean mayor a un número.

Tips:

- Utilizar al cargar el programa.

Uso:

```
ruby filtro_procesos.rb 250
```

Imagen 6.
Fuente: Desafío Latam.

Debe generar el archivo procesos_filtrados.data con:

```
299
312
412
```

Imagen 7.
Fuente: Desafío Latam.

Tips:

- Puedes ocupar los datos del archivo `procesos.data` como base para crear tu archivo.
- En la corrección el archivo contendrá distintos datos al presentado.
- La revisión se realizará sobre el archivo generado, este tiene que generarse en el mismo directorio de trabajo.