EXAMEN A DWEC 11-11-22 ALUMNE:.....

<u>1 punto</u> por la claridad y presentación del código del ejercicio, comentarios en el código y su indentación.

1. Simón dice (1p)

Detecta y corrige los errores del siguiente código para que el juego siga las siguientes especificaciones:

- El programa debe mostrar una secuencia de colores, agregando por cada ronda que pase, otro color más.
- Los colores posibles son: rojo, azul, verde y amarillo.
- El usuario debe ingresar los colores de la secuencia uno por uno.
- Si erra la secuencia (es decir, si ingresa algún color mal), el programa debe terminar mostrado el número de rondas que se sobrevivió.

Añade comentarios al código en las líneas donde veas "//"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Simon says</title>
</head>
<body>
<script>

const colors = ['blue’, 'red', 'yellow', 'green'];
const sequence = [];
let endGame = false;

while(!endGame) {
  //
  const random = Math.round((colors.length) * Math.random());
  const color = colors[random];
  //
  sequence.pop(color);
  //
  alert("Round ${sequence.length}. Current sequence is number : ${sequence}");
  for(let i = 1; i < sequence.length; i++) {
      if (!endGame) {
        const play = prompt(`Enter a color (${colors}):`);
        if (play !== sequence[i]) {
          endGame = true;
        }
      }
  }
}

alert(`You are wrong! You've survived ${sequence.length - 1} rounds`);
</script>
</body>
</html>
```

2. Array (1p)

/*Dado un array (puedes inicializar tu propio array en el código como en el ejemplo), quitar los elementos repetidos
El concepto puede ser fácilmente generalizado para tareas en el mundo real.
Por ejemplo: si necesitas esclarecer estadísticas removiendo elementos de baja frecuencia (ruido).
Deberás utilizar almenos una función que compruebe por cada elemento si está o no repetido en el array.
*/

Ejemplo:

```
const years= [2019, 2020, 2023, 2020, 2021, 2020]
norepetidos(years) // [2019, 2023, 2021]
```

3. Recursivo (1p)

Escribe una función recursiva que dado un número entero n, retorne un array con todos los números enteros en orden decreciente desde n a 1.
Escribe un programa donde pidas cuántos números queremos que contenga el array, llame a la función anterior y devuelva el array generado.

4. every (1p)

Crear una función **every** que acepte un array y un callback y que:

- por cada elemento del array ejecute el callback pasándole dicho elemento como argumento
- devuelva `true` si *todas* las llamadas al callback devolvieron `true`

Ejemplo:

```
const numeros = [10, 2, 3, 40, 33, 50]
const multiploDe10 = x => x % 10 === 0
const esPositivo = x => x >= 0

every(numeros, multiploDe10) // false
every(numeros, esPositivo) // true
```

5. Dados (2p)

Queremos programar un juego de dados en una página web.

- El usuario empieza con 50 euros para poder apostar
- Selecciona su apuesta
- Se "lanza" un dado (asignación al azar de un número del 1 al 6)
- Si el usuario acierta el número, gana el doble de lo que ha apostado. Si no acierta, pierde todo lo que ha apostado
- El juego acaba cuando el usuario llega a 0 euros o al llegar a 200 euros.

6. El ahorcado (3p)

Crear el juego del ahorcado siguiendo las especificaciones siguientes:

- El programa debe contar ya con una lista de posibles palabras.
- Debe mostrar inicialmente la palabra elegida aleatoriamente oculta con asteriscos, e ir preguntando por letras.

- A medida que se aciertan letras que contenga la palabra, se debe mostrar la palabra con las letras descubiertas.
- También se puede ingresar una palabra, pero si no se la adivina se pierde el juego.
- Si se adivinan todas las letras de la palabra, o se acierta la palabra, se gana.
- Si se intenta adivinar la palabra, pero se equivoca, o se ingresan 6 letras erróneas, se pierde.

Debes utilizar funciones para implementar el código.

Ejemplo:

```
// La palabra es ***********. Te quedan 6 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "a"
// La palabra es **********a*. Te quedan 6 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "e"
// La palabra es e*e*******a*. Te quedan 6 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "i"
// La palabra es e*e***i*i*a*. Te quedan 6 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "m"
// La palabra es e*e***i*i*a*. Te quedan 5 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "s"
// La palabra es e*e***i*i*a*. Te quedan 4 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "c"
// La palabra es e*ec**ici*a*. Te quedan 4 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "d"
// La palabra es e*e***i*idad. Te quedan 4 equivocaciones posibles. Ingresa una letra o intenta
adivinar la palabra:
// "electricidad"
// Felicitaciones, has adivinado la palabra.
```