EXAMEN B 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 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 = ['red', 'blue', 'yellow', 'green'];
const sequence = [];
let endGame = true;

while(!endGame) {
  //
  const random = Math.round(Math.random() * (colors.length));
  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 = false;
        }
      }
  }
}

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 numeros = [1, 2, 3, 3, 4,3]
repetidos(numeros) // [1, 2, 4]
```

3. Recursivo (1p)

Escribe una función granizo que tome un número entero y luego use console.log para mostrar la secuencia Hailstone para ese número, seguida de una línea mostrando el número de pasos tomados para llegar a 1. Por ejemplo, su programa debería poder

- para producir una ejecución de muestra que se vea así:
- Elige un número entero positivo y llámelo n.
- Si n es par, lo dividimos por dos.
- Si n es impar, lo multiplicamos por tres y le sumamos uno.
- Continúa este proceso hasta que n sea igual a uno.
- Si empezamos por el número 15, el resultado será:

```
15  is odd, so I make 3n+1:  46
46  is even, so I take half:  23
23  is odd, so I make 3n+1:  70
70  is even, so I take half:  35
35  is odd, so I make 3n+1:  106
106  is even, so I take half:  53
53  is odd, so I make 3n+1:  160
160  is even, so I take half:  80
80  is even, so I take half:  40
```

4. map (1p)

Crear una función **`map`** que acepte un array y una función callback y que:

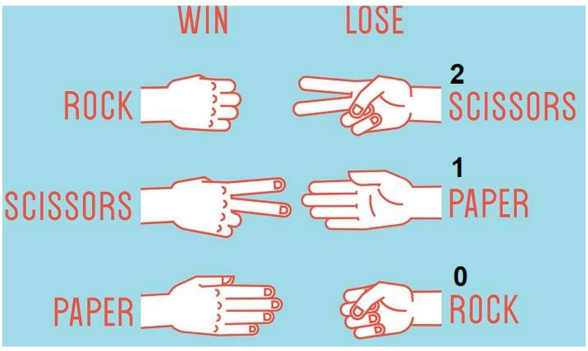
- por cada elemento del array ejecute el callback pasándole dicho elemento como argumento
- el resultado del callback lo inserte en un nuevo array
- devuelva el array final como resultado

Ejemplo:

```
const numeros = [1, 2, 3]
const duplicar = x => x * 2
map(numeros, duplicar) // [2, 4, 6]
```

5. Piedra, papel o tijera (2p)

Queremos programar el juego de piedra, papel o tijera en una página web (1 usuario contra el programa).



Sugerencia de codificación piedra=0; papel=1; tijera=2

- El usuario empieza con 50 puntos
- Selecciona su apuesta.
- Elije una opción (piedra, papel o tijera)
- El programa elige su opción (piedra, papel o tijera).
- Si el usuario gana, gana el doble de lo que ha apostado. Si pierde, pierde todo lo que ha apostado. En caso de empate, no pierde ni gana nada. (Mostrará mensaje del resultado de cada jugada)
- Se repite el juego hasta que el usuario llega a 0 euros o al llegar a 200 euros. En ambos casos, se devuelve un mensaje informando que ha acabado.

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:
```