RETO #1

Tema: ejercicios de lógica

Indicaciones:

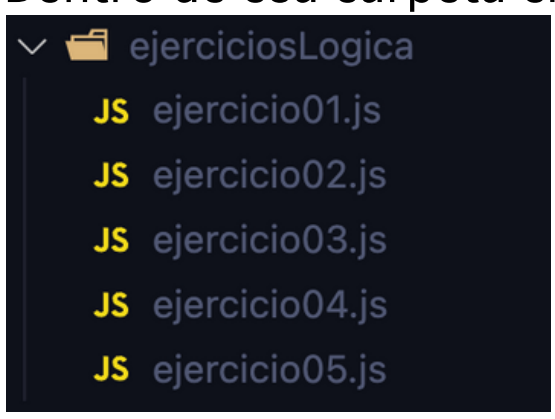
1. Crea en Trello un ticket por cada ejercicio y ve **moviéndolo de acuerdo al estado de desarrollo** en que esté (Crea SOLO un ticket a la vez).

2. Crea un repositorio en GitHub, el cual debes nombrar así:

- devTools-Bootcamp2023
  - Recuerda proteger la rama *"main"*
  - Recuerda solicitar la revisión en GitHub de tu tutor

3. Los ejercicios deberás trabajarlos en Visual Studio Code (VSC):

- Crea una carpeta
- Dentro de esa carpeta crea **un archivo por ejercicio**



4. En el repositorio de GitHub crea una branch llamada:

- tuNombre-Sprint1-Reto
  - Cada que termines un ejercicio crea un pull request (PR) a tu rama protegida y espera la aprobación de tu tutor.

Ejercicio #1

Número palíndromo

Un número palíndromo es aquel que se lee de la misma manera de izquierda a derecha y de derecha a izquierda.

En otras palabras, es un número que se mantiene igual cuando sus dígitos son invertidos.
Ejemplos: 121, 12321 y 333.

En este desafío, se te pide que crees una función llamada **"next_pal()"** que reciba como parámetro un número entero positivo, tu objetivo es que la función retorné el siguiente número palíndromo a partir del recibido por parámetro.

Aquí tienes algunos ejemplos:

- next_pal(11) debe devolver 22, ya que 22 es el siguiente número palíndromo después de 11.
- next_pal(188) debe devolver 191, ya que 191 es el siguiente número palíndromo después de 188.
- next_pal(191) debe devolver 202, ya que 202 es el siguiente número palíndromo después de 191.
- next_pal(2541) debe devolver 2552, ya que 2552 es el siguiente número palíndromo después de 2541.

Ejercicio #2

Crear un número de teléfono

Escribe una función que acepte un array con 10 números enteros positivos (entre 0 y 9) y devuelva una cadena de texto con esos números en forma de número de teléfono.

Ejemplo:

- createPhoneNumber([5, 5, 5, 1, 3, 4, 7, 8, 9, 0]) => devuelve "(555) 134-7890"
- createPhoneNumber([3, 0, 5, 1, 2, 7, 7, 2, 4, 9]) => devuelve "(305) 127-7249"

El formato del número de teléfono devuelto debe ser correcto para completar este desafío. ¡No olvides el espacio después del paréntesis de cierre!

Ejercicio #3

Contador de existencias

Dado un array de nombres, tu objetivo es crear una función que cuente cuántas veces se repite cada nombre y los represente con asteriscos (*).

Debes implementar la función **countNameRepetitions(names)**, que tomará como parámetro un array de nombres y devolverá un objeto con los nombres y su representación de asteriscos correspondiente.

Ejemplo:

```
const nombres = ['Juan', 'María', 'Pedro', 'Juan', 'María', 'María'];

// Llamada a la función
const resultado = countNameRepetitions(nombres);

// Resultado esperado
//   Juan: '**',
//   María: '***',
//   Pedro: '*'
```

En el ejemplo anterior, se tiene un array de nombres donde 'Juan' se repite dos veces, 'María' se repite tres veces y 'Pedro' se repite una vez.

La función **countNameRepetitions** debe imprimir los nombres y su representación de asteriscos correspondiente.

Recuerda que cada vez que un nombre se repite, se debe agregar un asterisco adicional al valor asociado al nombre en el objeto de resultado.

Ejercicio #4

FizzBuzz

Tu tarea es escribir un programa que recorra los números del 1 al n. Para cada número:

Si el número es divisible por 3, debes imprimir "Fizz".
Si el número es divisible por 5, debes imprimir "Buzz".
Si el número es divisible tanto por 3 como por 5, debes imprimir "FizzBuzz".
Si el número no es divisible ni por 3 ni por 5, debes imprimir el número tal cual.

Para lograrlo debes implementar una función llamada fizzBuzz(n), que tome como parámetro un número entero n y ejecute el juego FizzBuzz.

Ejemplo:

```
// Llamada a la función
fizzBuzz(15);

// Resultado esperado
// 1
// 2
// Fizz
// 4
// Buzz
// Fizz
// 7
// 8
// Fizz
// Buzz
// 11
// Fizz
// 13
// 14
// FizzBuzz
```

En el ejemplo anterior, la función fizzBuzz se ejecuta con el número 15. El programa recorre los números del 1 al 15 y aplica las reglas del juego FizzBuzz. Los números que son divisibles por 3 se reemplazan por "Fizz", los números divisibles por 5 se reemplazan por "Buzz" y los números divisibles tanto por 3 como por 5 se reemplazan por "FizzBuzz". El resultado se imprime en la consola.

Tu objetivo es implementar la función fizzBuzz de manera que cumpla con las reglas del juego y produzca el resultado esperado.

Ejercicio #5

Dibujando una X con asteriscos

Tu tarea es escribir un programa que dibuje una X utilizando asteriscos (*). La X debe tener un tamaño variable según el número ingresado.

Debes implementar una función llamada **drawX(tamano)**, donde el parámetro tamano es un número entero y dibuje la X correspondiente.

Ejemplo:

```
// Llamada a la función
dibujarX(5);
// Resultado esperado
// *   *
//  * *
//   *
//  * *
// *   *
```

En el ejemplo anterior, la función drawX() se ejecuta con el tamaño 5. El programa dibuja una X formada por asteriscos, donde el tamaño se adapta según el número ingresado.
En este caso, se muestra una X con dimensiones 5x5.

Tu objetivo es implementar la función drawX() de manera que genere correctamente la X con asteriscos de acuerdo al tamaño especificado.

softserve

devtools.academypro@gmail.com