1) Utilizando función arrow, crear una función que reciba como parámetros un nombre, apellido y edad y los retorne en un string concatenado “Hola mi nombre es sebastián yabiku y mi edad 33”
2) Cree una función que tome números y devuelva la suma de sus cubos.

   sumOfCubes(1, 5, 9) ➞ 855

   ```
   // Since 1^3 + 5^3 + 9^3 = 1 + 125 + 729 = 855
   ```
3) Crear una funcion que me retorne el tipo de valor entregado, invocar la función para los distintos tipos de js
4) Crear una función que reciba n cantidad de argumentos y los sume (utilizar parametros rest)
5) Crear una función que reciba un array de valores y filtre los valores que no son string
6) Cree una función que tome una matriz de números y devuelva los números mínimos y máximos, en ese orden.

   minMax([1, 2, 3, 4, 5]) ➞ [1, 5]

7) Escriba una función que tome una matriz de 10 enteros (entre 0 y 9) y devuelva una cadena en forma de un número de teléfono.

   formatPhoneNumber([1, 2, 3, 4, 5, 6, 7, 8, 9, 0]) ➞ "(123) 456-7890"

8) Cree una función que tome una matriz de matrices con números. Devuelve una nueva matriz (única) con el mayor número de cada uno.

   findLargestNums([[4, 2, 7, 1], [20, 70, 40, 90], [1, 2, 0]]) ➞ [7, 90, 2]

9) Dada una palabra, escriba una función que devuelva el primer índice y el último índice de un carácter.

   charIndex("hello", "l") ➞ [2, 3]

   ```
   // The first "l" has index 2, the last "l" has index 3.
   ```

   charIndex("circumlocution", "c") ➞ [0, 8]

   ```
   // The first "c" has index 0, the last "c" has index 8.
   ```

10) Escriba una función que convierta un objeto en una matriz, donde cada elemento representa un par clave-valor.

    toArray({ a: 1, b: 2 }) ➞ [["a", 1], ["b", 2]]

11) Cree la función que toma una matriz con objetos y devuelve la suma de los presupuestos de las personas.

```
getBudgets([
  { name: "John", age: 21, budget: 23000 },
  { name: "Steve",  age: 32, budget: 40000 },
  { name: "Martin",  age: 16, budget: 2700 }
]) → 65700
```

12) Cree una función que tome una matriz de estudiantes y devuelva una matriz de nombres de estudiantes.

```
getStudentNames([
  { name: "Steve" },
  { name: "Mike" },
  { name: "John" }
]) → ["Becky", "John", "Steve"]
```

13) Escriba una función que convierta un objeto en una matriz de claves y valores.

```
objectToArray({
  likes: 2,
  dislikes: 3,
  followers: 10
}) → [["likes", 2], ["dislikes", 3], ["followers", 10]]
```

14) Cree una función donde, dado el número n, devuelva la suma de todos los números cuadrados incluyendo n.

squaresSum(3) → 14

```
// 1² + 2² + 3² =
// 1 + 4 + 9 =
// 14
```

15) Cree una función para multiplicar todos los valores en una matriz por la cantidad de valores en la matriz dada

multiplyByLength([2, 3, 1, 0]) → [8, 12, 4, 0]

16) Cree una función que tome un número como argumento y devuelva una matriz de números contando desde este número a cero.
countdown(5) ➞ [5, 4, 3, 2, 1, 0]

17) Cree una función que tome una matriz y devuelva la diferencia entre los números más grandes y más pequeños.
diffMaxMin([10, 4, 1, 4, -10, -50, 32, 21]) ➞ 82
// Smallest number is -50, biggest is 32.

18) Cree una función que filtre las cadenas de una matriz y devuelva una nueva matriz que solo contenga enteros.
filterList([1, 2, 3, "x", "y", 10]) ➞ [1, 2, 3, 10]

19) Cree una función que tome dos argumentos (elemento, tiempos). El primer argumento (elemento) es el elemento que necesita repetirse, mientras que el segundo argumento (veces) es la cantidad de veces que se debe repetir el elemento. Devuelve el resultado en una matriz.
repeat(13, 5) ➞ [13, 13, 13, 13, 13]

20) Escriba una función, .vreplace () que extienda el prototipo de cadena reemplazando todas las vocales en una cadena con una vocal especificada.
"apples and bananas".vreplace("u") ➞ "upplus und bununus"

21) Te dan una cadena de palabras. Debe encontrar la palabra "Nemo" y devolver una cadena como esta: "¡Encontré a Nemo en [el orden de la palabra que encuentra nemo]!".
findNemo("I am finding Nemo !") ➞ "I found Nemo at 4!"

22) Cree una función que capitalice la última letra de cada palabra.
capLast("hello") ➞ "hellO"