1. Fix Error: manipulación de arreglos

¡Ayuda a corregir todos los errores en la función `incrementItems`! ¡Está destinado a agregar 1 a cada elemento en el arreglo!

```
function incrementItems(arr) {
  for (let i = 0; i < array.length; i++) {
    arr[i] === arr[i] + 1
  }

  return array
}
```

Test Case	Expected
incrementItems([0, 1, 2, 3])	[1, 2, 3, 4]
incrementItems([2, 4, 6, 8])	[3, 5, 7, 9]
incrementItems([-1, -2, -3, -4])	[0, -1, -2, -3]

2. Fix Error: Valor vs. Referencia de Tipos

Cree una función que devuelva `true` si dos arreglos contienen valores idénticos y `false` en caso contrario.

Para resolver esta pregunta, tu amigo escribe el siguiente código:

```
function checkEquals(arr1, arr2) {
if (arr1 === arr2) {
  return true
 } else {
  return false
 }
}
```

Pero probando el código, ves que algo no está del todo bien. Ejecutar el código arroja los siguientes resultados:

```
checkEquals([1, 2], [1, 3]) → false
// Good so far...
```

```
checkEquals([1, 2], [1, 2]) → false
// Yikes! What happened?
```

Reescribe el código de tu amigo para que verifique correctamente si dos arreglos son iguales. Para ser iguales, los arreglos deben tener los mismos elementos en el mismo orden.

Las siguientes pruebas deben pasar:

Test Case	Expected
checkEquals([1, 2], [1, 3])	`false`
checkEquals([1, 2], [1, 2])	`true`
checkEquals([4, 5, 6], [4, 5, 6])	`true`
checkEquals([4, 7, 6], [4, 5, 6])	`false`
checkEquals([4, 7, 6], [4, 6, 7])	`false`

3. Comprobar si la propiedad existe en el objeto

Escriba una función que tome un objeto (a) y un string (b) como argumento. Devuelva `true` si el objeto tiene una propiedad con la clave 'b'. Devuelva `false` de lo contrario.

```
function myFunction(a, b) {

  return
}
```

Test Case	Expected
myFunction({a:1,b:2,c:3},'b')	`true`

myFunction({x:'a',y:'b',z:'c'},'a')	`false`
myFunction({x:'a',y:'b',z:undefined},'z')	`false`

4. Votos a favor vs votos en contra

Escribir una función que tome un objeto como argumento. El objeto contiene dos propiedades, `upvotes` y `downvotes`. Devuelve el número de votos a favor menos el número de votos en contra.

```
function getVoteCount(obj) {

  return
}
```

Test Case	Expected
getVoteCount({upvotes:13, downvotes:0})	13
getVoteCount({upvotes:2, downvotes:33})	-31
getVoteCount({upvotes:132, downvotes:132})	0