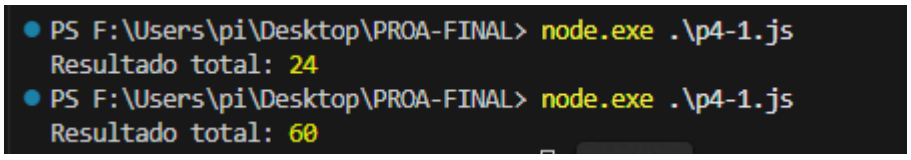
Ejercicio 1: Probar el código

```
// Variable global que indica quién tiene el asiento
var elAsiento = "nadie";

// Esta función simula que el asiento cambia de dueño después de un tiempo
function cambiarNombre(nombre) {
  setTimeout(function () {
    console.log(" *** elAsiento es para: " + nombre + " ***");
    elAsiento = nombre; // Aquí es donde se cambia el nombre
  }, 100); // espera 100 ms antes de ejecutar
}

// Esta función intenta reservar el asiento
function hacerReserva(nombre) {
  if (elAsiento == "nadie") {
    cambiarNombre(nombre); // Cambia el dueño si está libre
  }
}

// Simulación de dos reservas
console.log("Intento reservar para juan.");
hacerReserva("juan");

console.log("Intento reservar para pepe.");
hacerReserva("pepe");

// Mostramos el valor final del asiento después de 1 segundo
setTimeout(() => console.log("elAsiento finalmente es para " + elAsiento), 1000);
```

```
PS F:\Users\pi\Desktop\PROA-FINAL> node.exe .\p4.js
Intento reservar para juan.
Intento reservar para pepe.
 *** elAsiento es para: juan ***
 *** elAsiento es para: pepe ***
elAsiento finalmente es para pepe
```

Ejercicio 2: Usar Promesas

```
function porDos(n, callback) {
  setTimeout(() => {
    callback(n * 5);
  }, 1000);
}

// Llamadas anidadas (pirámide fea 😵)
porDos(3, function (a) {
  porDos(4, function (b) {
    porDos(5, function (c) {
      console.log("Resultado total:", a + b + c);
    });
  });
});
```



La otra es *5

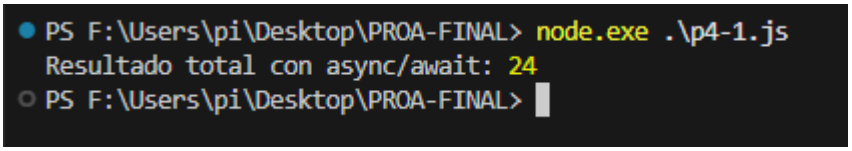
Promesas encadenadas

```
function porDos(n) {
  return new Promise((resolver, rechazar) => {
    setTimeout(() => {
      resolver(n * 2);
    }, 1000);
  });
}

// Encadenando promesas para que sea más legible
let a, b, c;
porDos(3)
  .then((r) => {
    a = r;
    return porDos(4);
  })
  .then((r) => {
    b = r;
    return porDos(5);
  })
  .then((r) => {
    c = r;
    return a + b + c;
  })
  .then((total) => {
    console.log("Resultado total con promesas:", total);
  });
```

```
PS F:\Users\pi\Desktop\PROA-FINAL> node.exe .\p4-1.js
Resultado total con promesas: 24
PS F:\Users\pi\Desktop\PROA-FINAL>
```

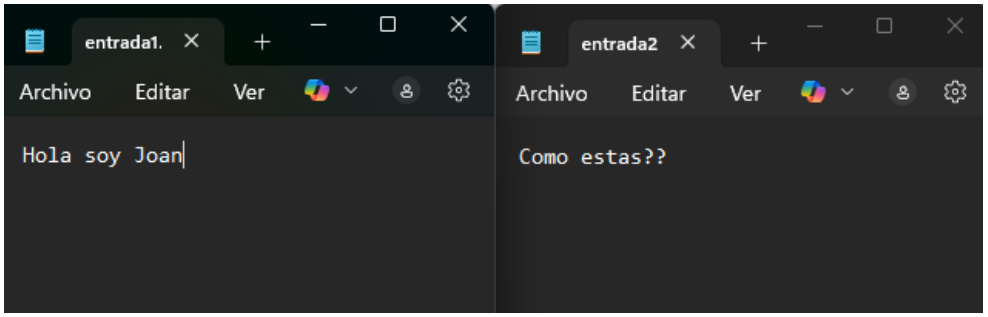
async/await

```
function porDos(n) {
  return new Promise((resolver, rechazar) => {
    setTimeout(() => {
      resolver(n * 2);
    }, 1000);
  });
}

async function hacerUnaSuma() {
  const a = await porDos(3); // espera a que termine
  const b = await porDos(4);
  const c = await porDos(5);
  return a + b + c;
}

// Llamamos a la función async y mostramos el resultado
hacerUnaSuma().then((total) => {
  console.log("Resultado total con async/await:", total);
});
```



Ejercicio 3: Leer y escribir ficheros con promesas

```
const fs = require("fs").promises;

// Lee  fichero y devuelve su contenido
function leerFichero(nombre) {
  return fs.readFile(nombre, "utf-8");
}

// Escribe en un fichero
function escribirFichero(nombre, contenido) {
  return fs.writeFile(nombre, contenido);
}

// Une el contenido de dos archivos y lo guarda en uno
async function concatenarFicheros(origen1, origen2, destino) {
  try {
    const contenido1 = await leerFichero(origen1);
    const contenido2 = await leerFichero(origen2);
    const combinado = contenido1 + "\n" + contenido2;

    await escribirFichero(destino, combinado);
    console.log("Todo bien mira en:", destino);
  } catch (error) {
    console.error("Error", error.message);
  }
}

// Prueba de la función:
concatenarFicheros("entrada1.txt", "entrada2.txt", "salida.txt");
```



```
PS F:\Users\pi\Desktop\PROA-FINAL> node.exe .\ConcatenarFicheros.js
Todo bien mira en: salida.txt
PS F:\Users\pi\Desktop\PROA-FINAL>
```