






# INSTRUCCIONES

## INFORMACIÓN IMPORTANTE:

1) Antes de empezar asegúrate de tener todo instalado correctamente:

-  Visual Studio Code
-  ANTLR4 que instalaremos desde Visual Studio Code
-  Node.js
-  Java
-  GitHub (también es posible abrirlo desde la web)

2) En mi index tengo inicializado estos dispositivos:

- `visitor.estadoDispositivos.set("sensor1", "encendido");`
- `visitor.estadoDispositivos.set("temperatura", 35);`
- `visitor.estadoDispositivos.set("bombaAgua", "apagado");`
- `visitor.estadoDispositivos.set("generador1", "apagado");`

Por lo tanto, mis inputs validos serian algo asi:

- cuando `sensor1` es encendido entonces activar `motor_1`;
- cuando `bombaAgua` es apagado entonces activar `generador1`;

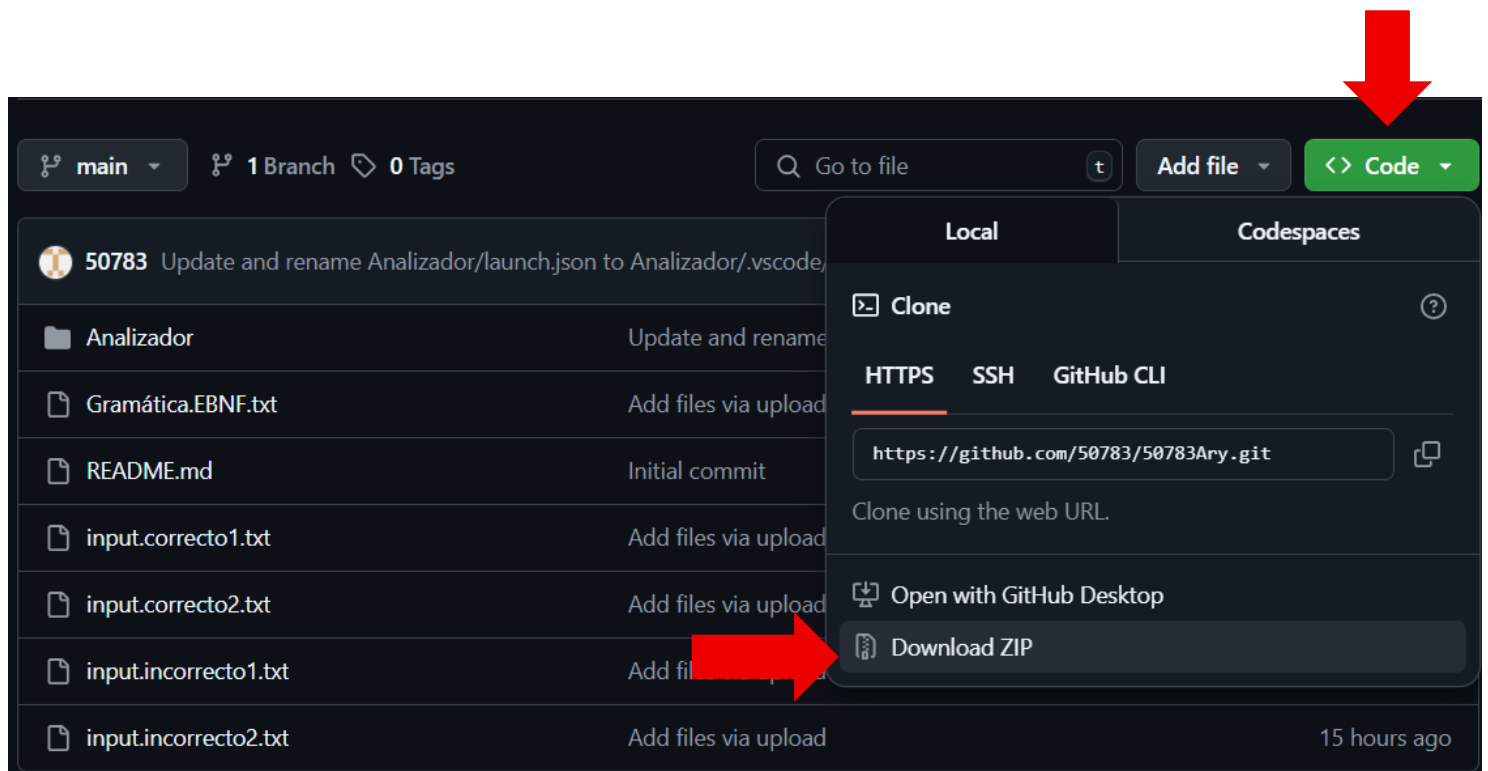
Al inicializar manualmente en el archivo `index.js` los dispositivos con valores específicos usando `visitor.estadoDispositivos.set(...)`, estoy definiendo el estado inicial del entorno simulado.

Esto permite que, al evaluar condiciones en el input, el intérprete pueda determinar si se deben ejecutar las acciones correspondientes.

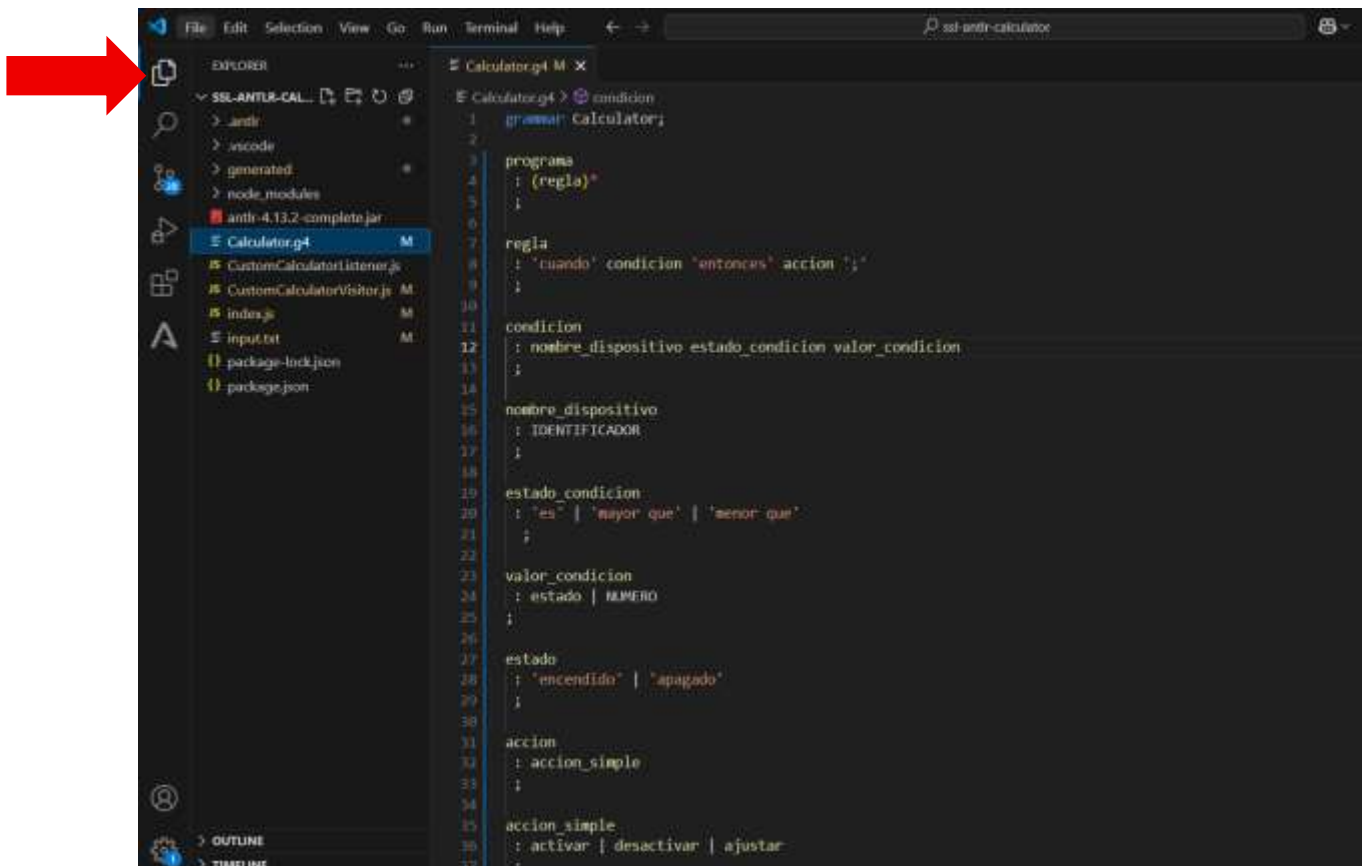
**Si introduzco un input que hace referencia a un dispositivo que no fue previamente inicializado, su estado será undefined, por lo que la condición no se cumplirá y la acción no se ejecutará, aunque el código sea sintácticamente válido.**

3) Para verificar que todo funcione es necesario poner en el terminal  
**`node index.js`**

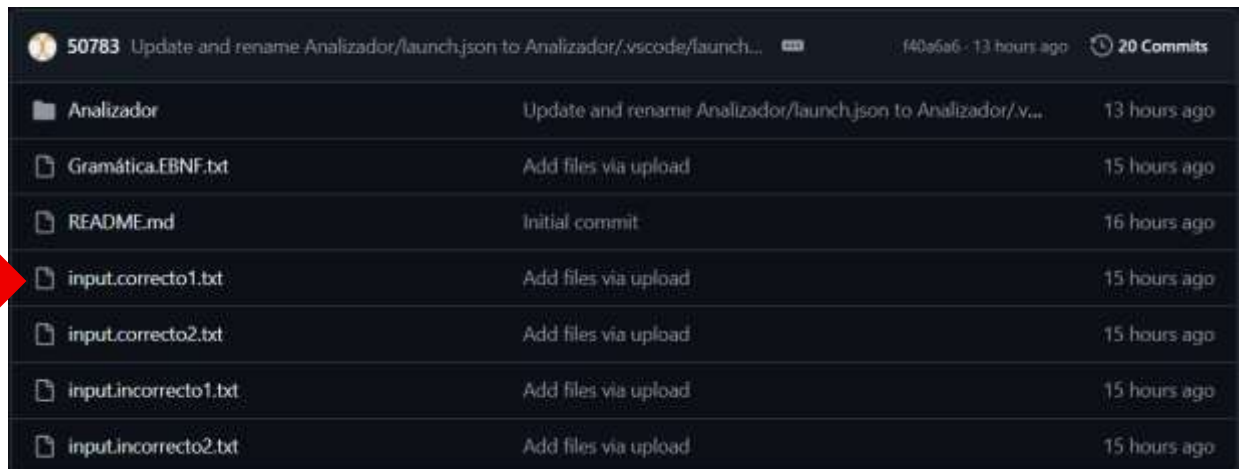
- 1) Entrar al repositorio de GitHub <https://github.com/50783/50783> y descargar



- 2) Una vez descargado correctamente, en Visual Studio Code, deberías ver lo siguiente en **Explorer**



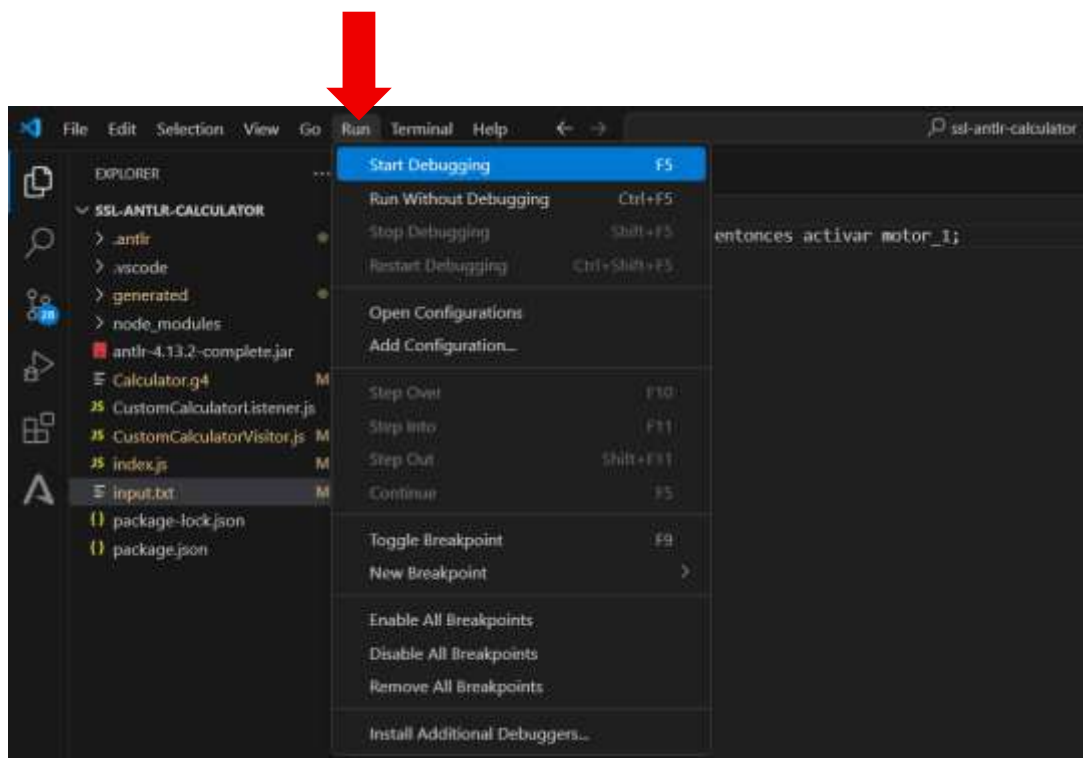
- 3) Ahora volvemos a GitHub <https://github.com/50783/50783> abrimos el **input.correcto1.txt** y **copiamos el texto dentro del documento (Ctrl + C)**

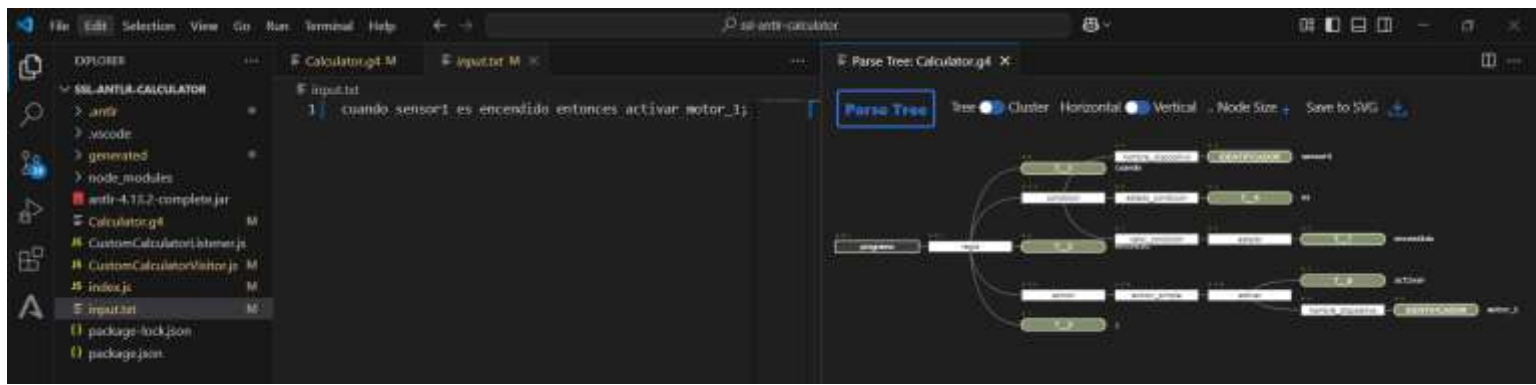


- 4) En Visual Studio Code, **input.txt** **pegamos el texto (Ctrl + V)**

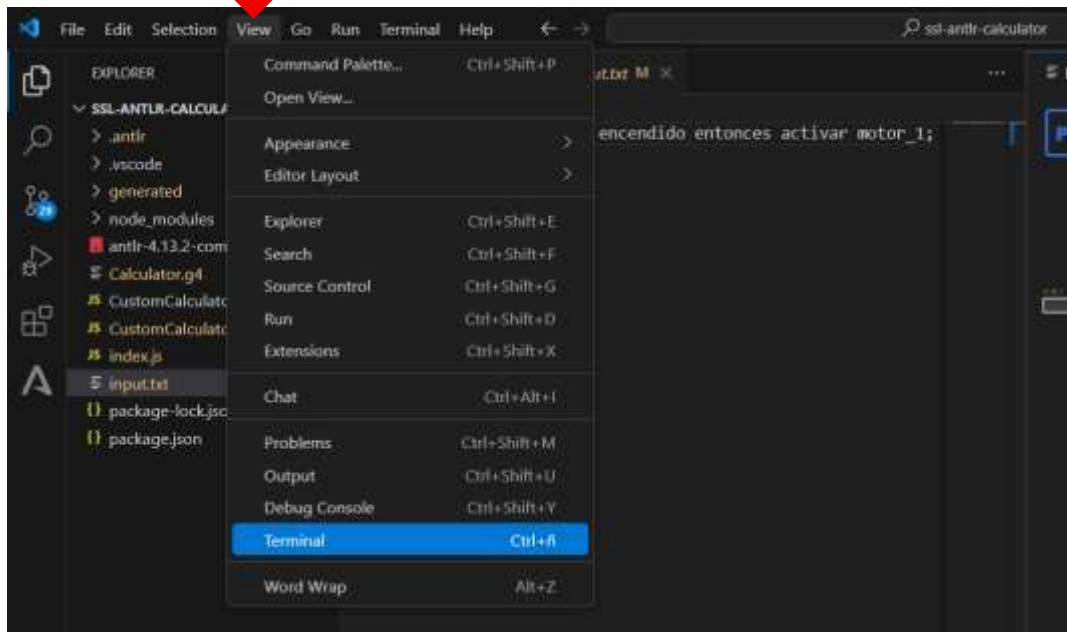


- 5) Es importante **Ejecutar – Depuración (Run - Start Debugging)** para lograr ver el **Parse Tree**





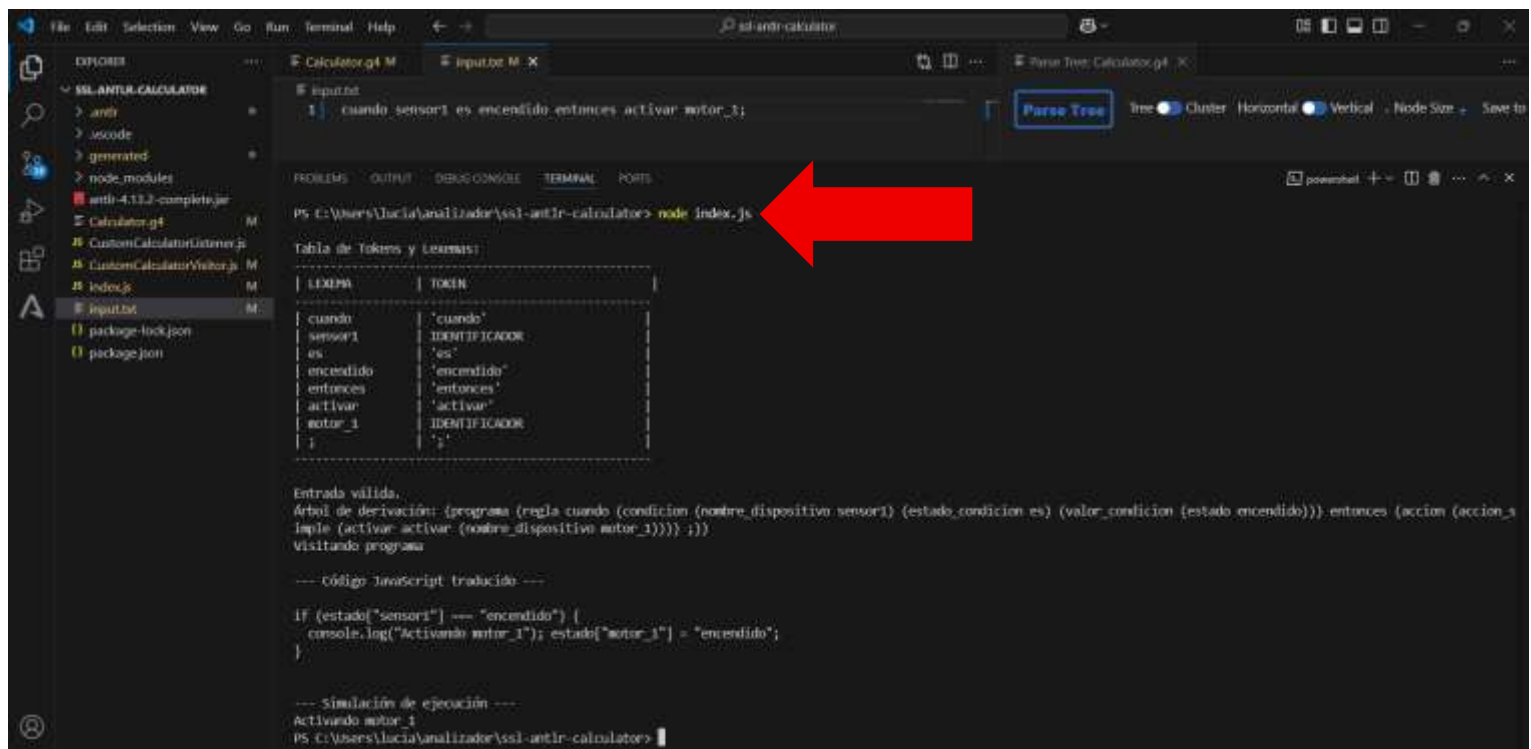
- 6) Una vez pegado el input.correcto1.txt y **Ejecutar – Depuración** abriremos el Terminal (**View – Terminal**)



7) En el terminal escribimos **node index.js** y damos **Enter**

En consecuencia, nos devuelve:

- Tabla de Tokens y Lexemas
- Árbol de Derivación
- Código de JavaScript Traducido



```
PS C:\Users\lucia\analizador\ssl-antlr-calculador> node index.js

Tabla de Tokens y Lexemas:
+-----+-----+
| LEXEMA | TOKEN |
+-----+-----+
| cuando | 'cuando' |
| sensor1 | IDENTIFICADOR |
| es | 'es' |
| encendido | 'encendido' |
| entonces | 'entonces' |
| activar | 'activar' |
| motor_1 | IDENTIFICADOR |
| ; | ';' |
+-----+-----+

Entrada válida.
Árbol de derivación: (programa (regla cuando (condicion (nombre_dispositivo sensor1) (estado_condicion es) (valor_condicion (estado encendido))) entonces (accion (accion_s
imple (activar activar (nombre_dispositivo motor_1)))));))
visitando programa

--- código javascript traducido ---

if (estado["sensor1"] === "encendido") {
  console.log("Activando motor_1"); estado["motor_1"] = "encendido";
}

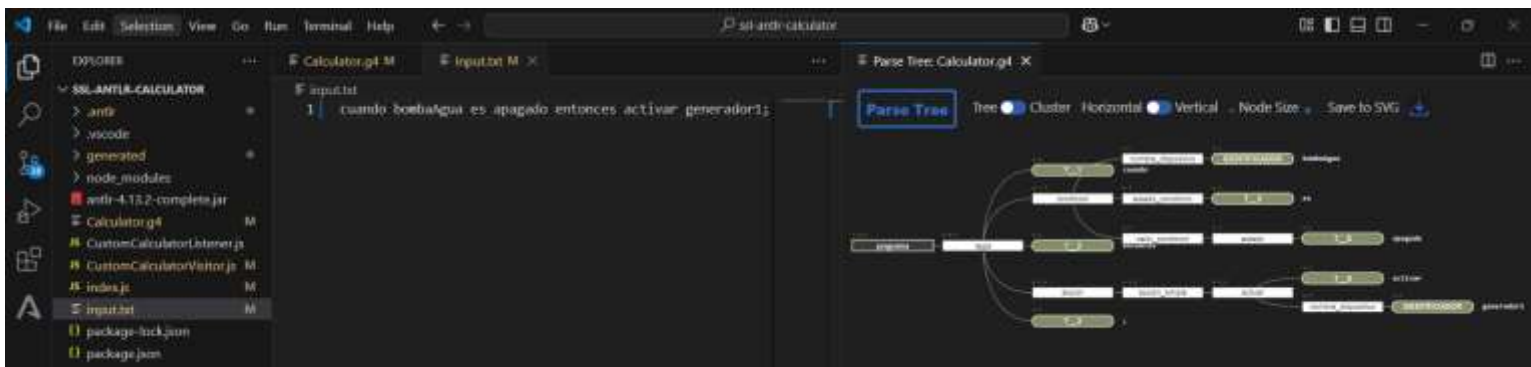
--- simulación de ejecución ---
Activando motor_1
PS C:\Users\lucia\analizador\ssl-antlr-calculador>
```

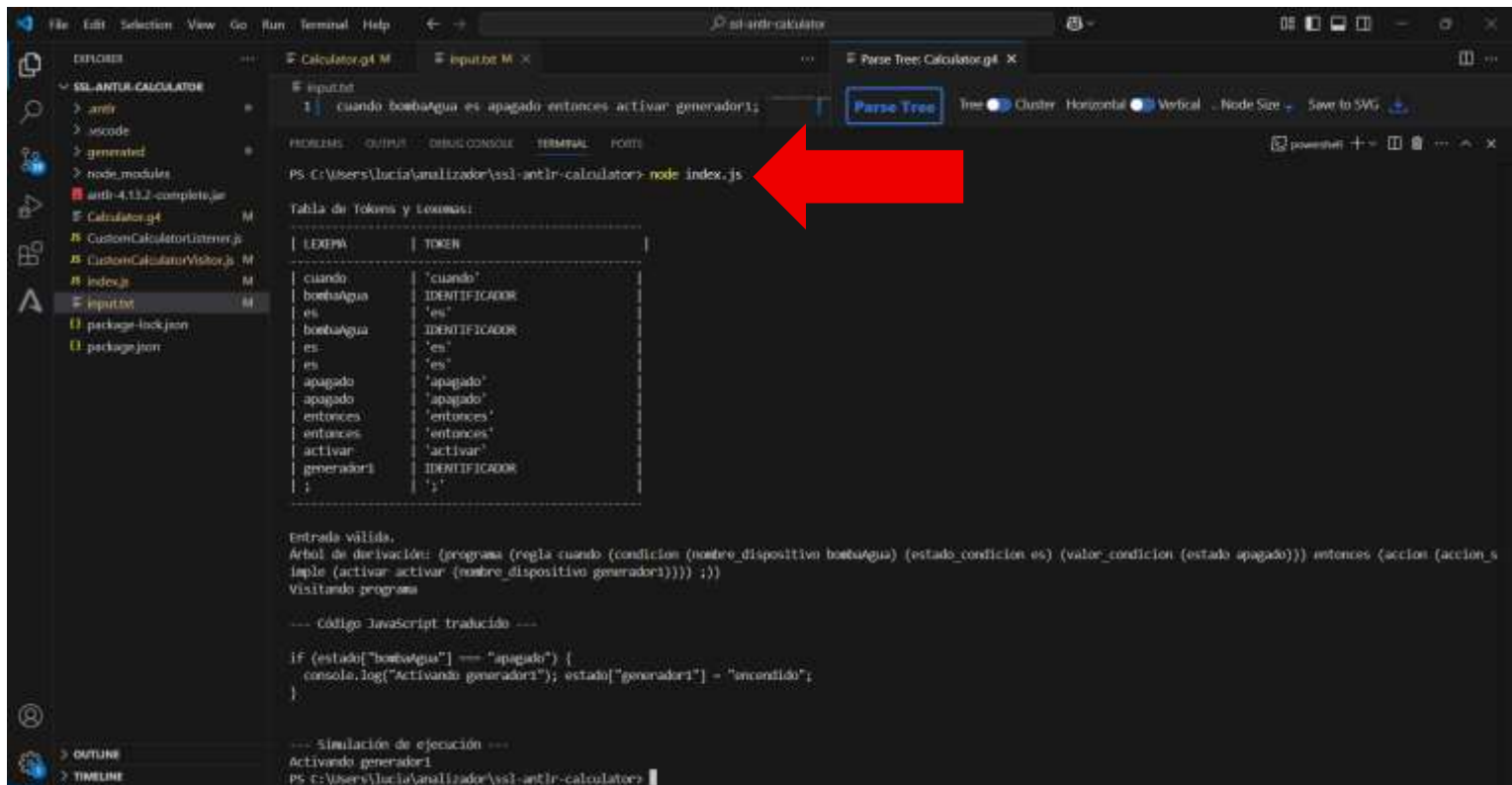
8) Ahora volvemos a GitHub <https://github.com/50783/50783> abrimos el **input.correcto2.txt**

Copiamos el texto dentro del documento (**Ctrl + C**) y

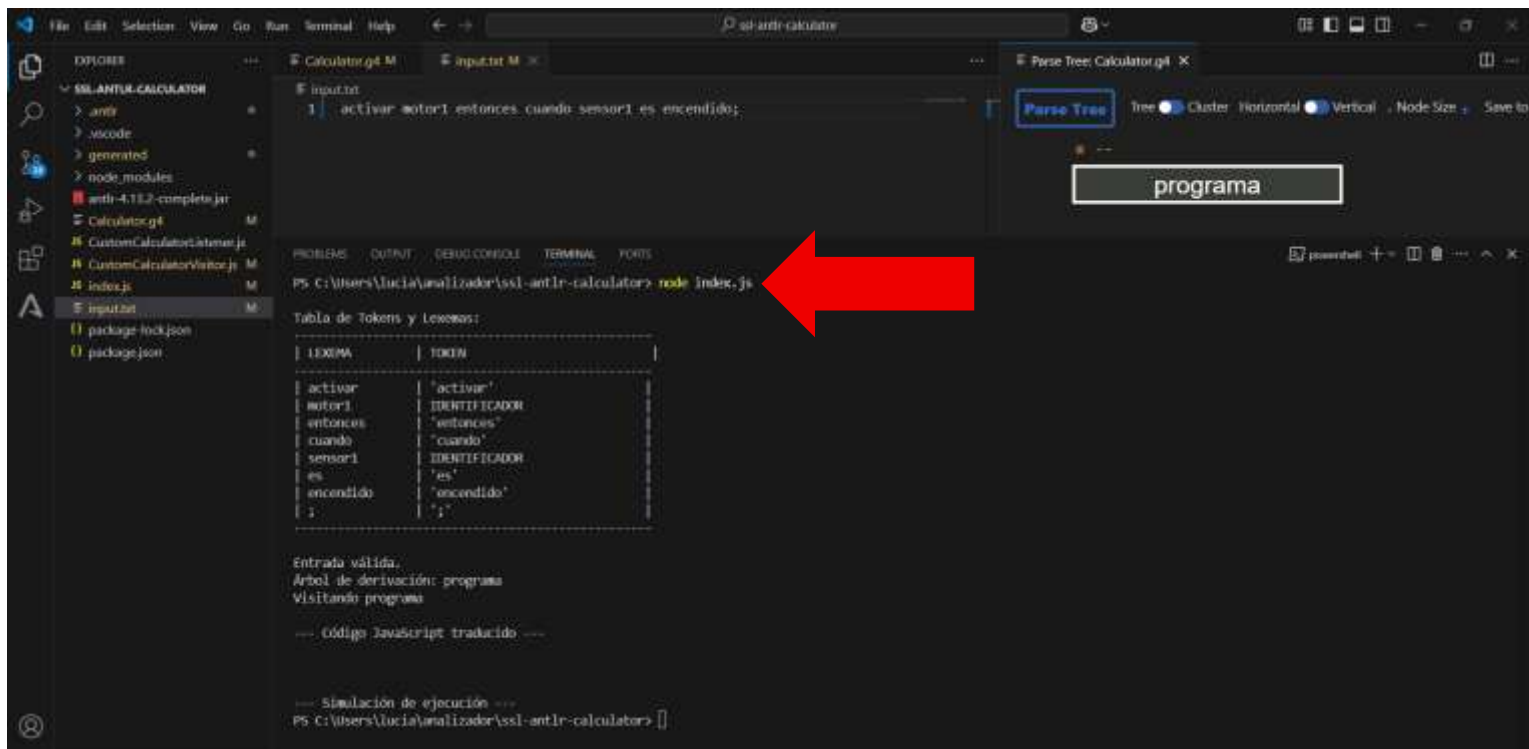
En Visual Studio Code, **input.txt** pegamos el texto (**Ctrl + V**)

*Realizamos lo mismo que vimos anteriormente*





9) Realizamos lo mismo con **input.incorrecto1.txt**





10) Realizamos lo mismo con **input.incorrecto2.txt**

The screenshot shows the Visual Studio Code interface. The Explorer view on the left shows the project structure for 'ssl-antlr-calculator'. The Editor view shows the 'input.txt' file with the text: `1 cuando 123sensor es encendido entonces activar motor1;`. The Parse Tree view on the right shows a tree structure for the input. The Terminal view at the bottom shows the command `node index.js` being executed, which results in a syntax error: `Line 1:7 extraneous input '123' expecting IDENTIFICACION`. A red arrow points to the terminal output.

Terminal Output:

```
PS C:\Users\lucia\analizador\ssl-antlr-calculator> node index.js

Tabla de Tokens y lexemas:

| LEXEMA | TOKEN |
|-----|-----|
| cuando | 'cuando' |
| 123 | NUMERO |
| sensor | IDENTIFICACION |
| es | 'es' |
| encendido | 'encendido' |
| entonces | 'entonces' |
| activar | 'activar' |
| motor1 | IDENTIFICACION |
| ; | ';' |

Line 1:7 extraneous input '123' expecting IDENTIFICACION

Se encontraron errores de sintaxis en la entrada.
PS C:\Users\lucia\analizador\ssl-antlr-calculator>
```