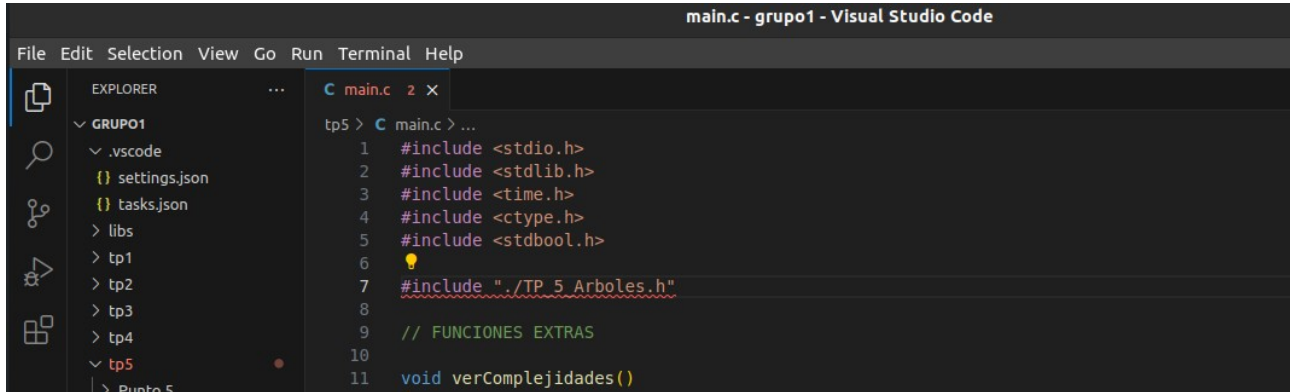


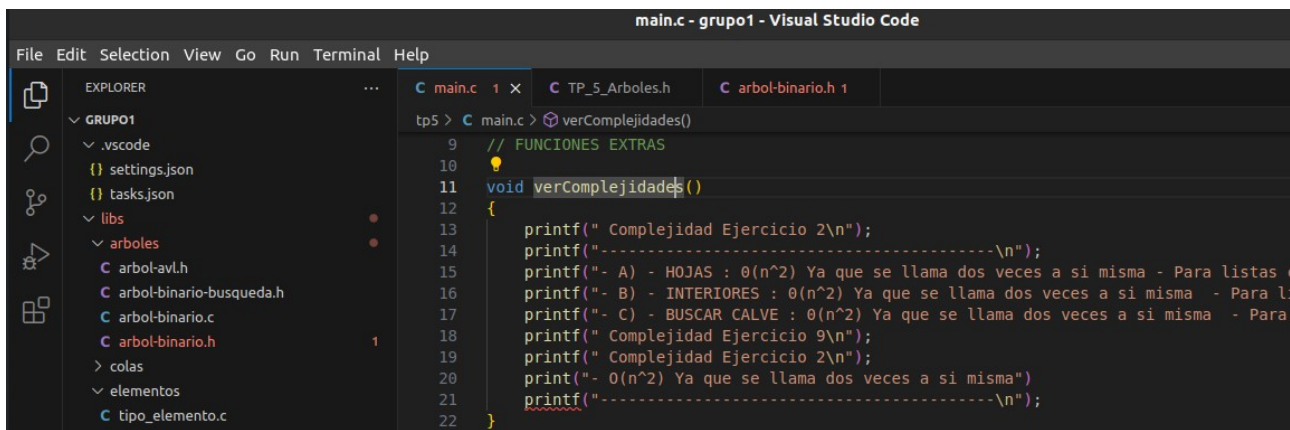
GRUPO 1 – Correcciones Trabajo Práctico: ÁRBOLES

RESULTADO DE LA CORRECCIÓN: **DESAPROBADO**

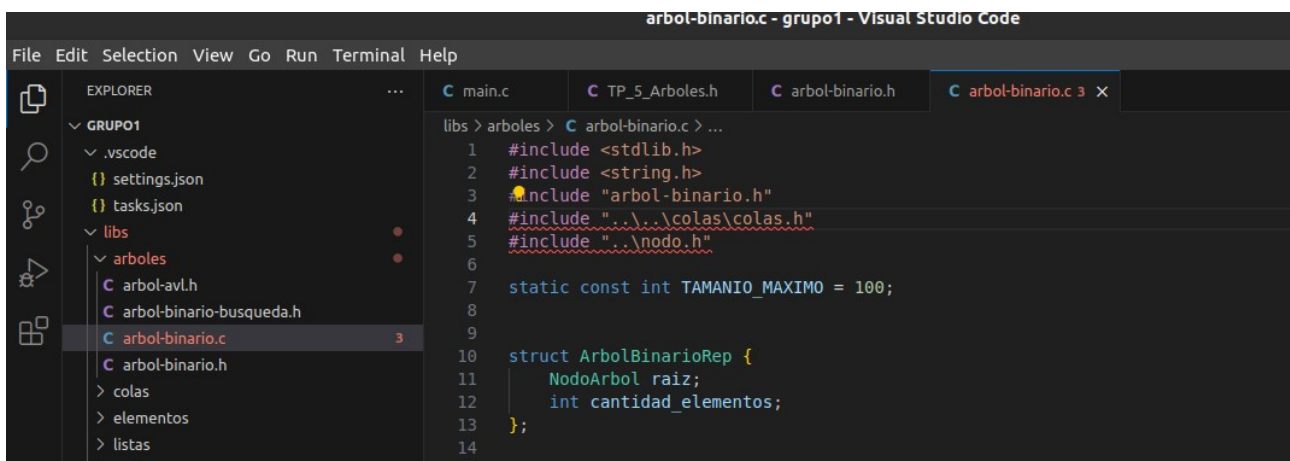
La mayoría de los includes están mal, no respetan las estructuras de los directorios. Hay errores de tipeo y de palabras mal escritas.



```
main.c - grupo1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
GRUPO1
  .vscode
  settings.json
  tasks.json
  libs
  tp1
  tp2
  tp3
  tp4
  tp5
  Punto 5
main.c 2 x
tp5 > C main.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <ctype.h>
5 #include <stdbool.h>
6
7 #include "../TP_5_Arboles.h"
8
9 // FUNCIONES EXTRAS
10
11 void verComplejidades()
```



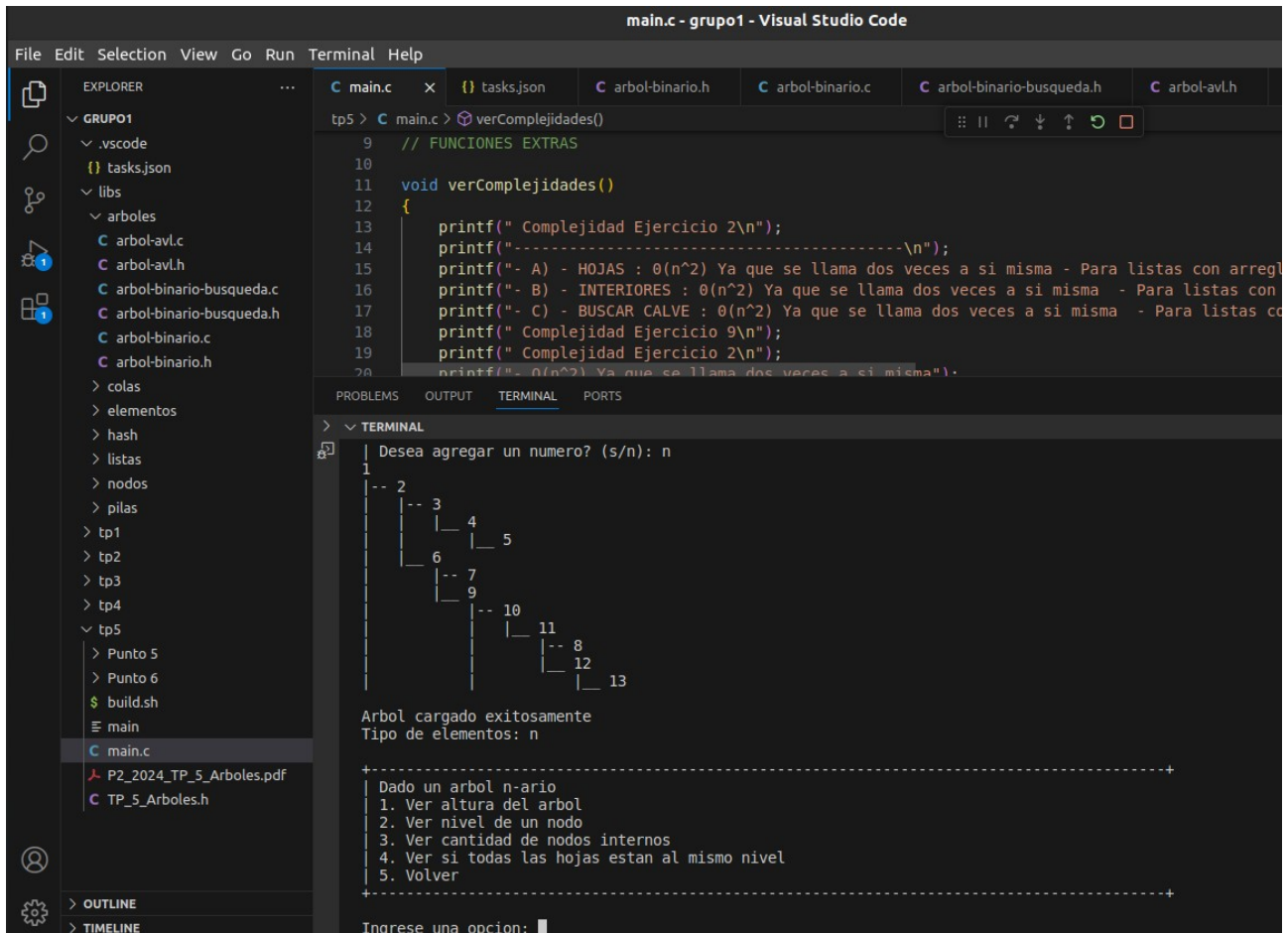
```
main.c - grupo1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
GRUPO1
  .vscode
  settings.json
  tasks.json
  libs
  arboles
  arbol-avl.h
  arbol-binario-busqueda.h
  arbol-binario.c
  arbol-binario.h
  colas
  elementos
  tipo_elemento.c
main.c 1 x TP_5_Arboles.h arbol-binario.h 1
tp5 > C main.c > verComplejidades()
9 // FUNCIONES EXTRAS
10
11 void verComplejidades()
12 {
13     printf(" Complejidad Ejercicio 2\n");
14     printf("-----\n");
15     printf("- A) - HOJAS : 0(n^2) Ya que se llama dos veces a si misma - Para listas d
16     printf("- B) - INTERIORES : 0(n^2) Ya que se llama dos veces a si misma - Para l
17     printf("- C) - BUSCAR CALVE : 0(n^2) Ya que se llama dos veces a si misma - Para
18     printf(" Complejidad Ejercicio 9\n");
19     printf(" Complejidad Ejercicio 2\n");
20     print("- 0(n^2) Ya que se llama dos veces a si misma")
21     printf("-----\n");
22 }
```



```
arbol-binario.c - grupo1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
GRUPO1
  .vscode
  settings.json
  tasks.json
  libs
  arboles
  arbol-avl.h
  arbol-binario-busqueda.h
  arbol-binario.c
  arbol-binario.h
  colas
  elementos
  listas
main.c TP_5_Arboles.h arbol-binario.h arbol-binario.c 3 x
libs > arboles > C arbol-binario.c > ...
1 #include <stdlib.h>
2 #include <string.h>
3 #include "arbol-binario.h"
4 #include "..\\colas\\colas.h"
5 #include "..\\nodo.h"
6
7 static const int TAMANIO_MAXIMO = 100;
8
9
10 struct ArbolBinarioRep {
11     NodoArbol raiz;
12     int cantidad_elementos;
13 };
14
```

Por cada elemento a ingresar piden la confirmación, muy complicado para probar. En el ejercicio 8 muestra un árbol e indica incorrectamente que la altura es 2.

GRUPO 1 – Correcciones Trabajo Práctico: ÁRBOLES



The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a project structure for 'GRUPO1'. The main editor shows the 'main.c' file with the following code:

```
tp5 > C main.c > verComplejidades()
9 // FUNCIONES EXTRAS
10
11 void verComplejidades()
12 {
13     printf(" Complejidad Ejercicio 2\n");
14     printf("-----\n");
15     printf("- A) - HOJAS :  $O(n^2)$  Ya que se llama dos veces a si misma - Para listas con arreglo\n");
16     printf("- B) - INTERIORES :  $O(n^2)$  Ya que se llama dos veces a si misma - Para listas con\n");
17     printf("- C) - BUSCAR CALVE :  $O(n^2)$  Ya que se llama dos veces a si misma - Para listas con\n");
18     printf(" Complejidad Ejercicio 9\n");
19     printf(" Complejidad Ejercicio 2\n");
20     printf("-  $O(n^2)$  Ya que se llama dos veces a si misma");
```

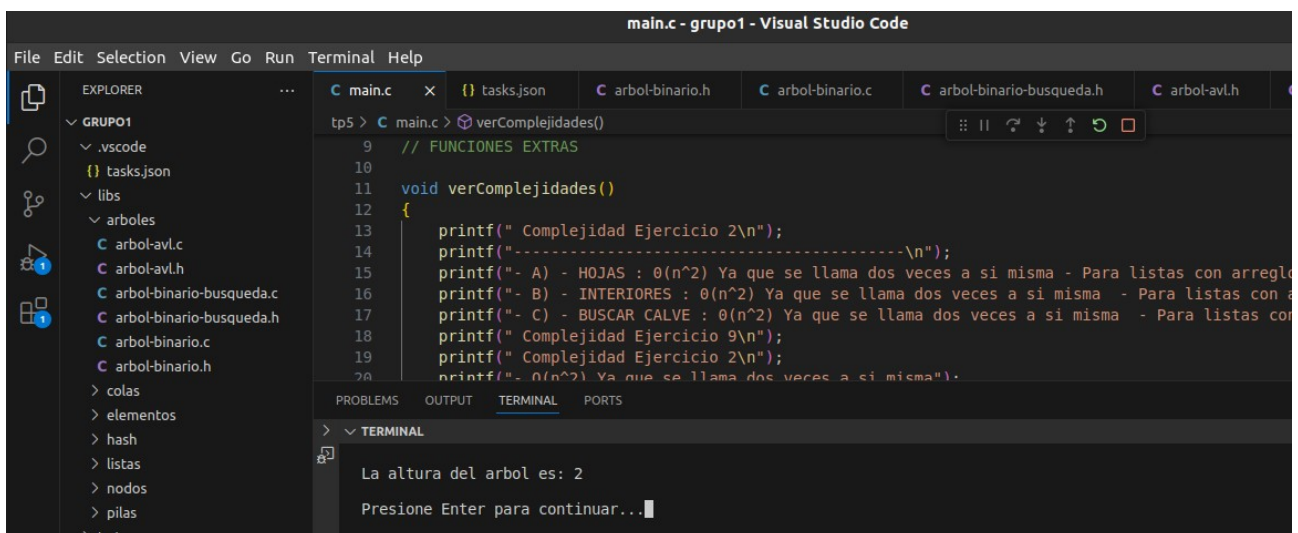
The terminal output shows the execution of the program:

```
Desea agregar un numero? (s/n): n
1
|-- 2
   |-- 3
       |-- 4
           |-- 5
       |-- 6
           |-- 7
               |-- 9
                   |-- 10
                       |-- 11
                           |-- 8
                               |-- 12
                                   |-- 13

Arbol cargado exitosamente
Tipo de elementos: n

+-----+
| Dado un arbol n-ario |
| 1. Ver altura del arbol |
| 2. Ver nivel de un nodo |
| 3. Ver cantidad de nodos internos |
| 4. Ver si todas las hojas estan al mismo nivel |
| 5. Volver |
+-----+
```

The user is prompted to enter an option.



The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a project structure for 'GRUPO1'. The main editor shows the 'main.c' file with the following code:

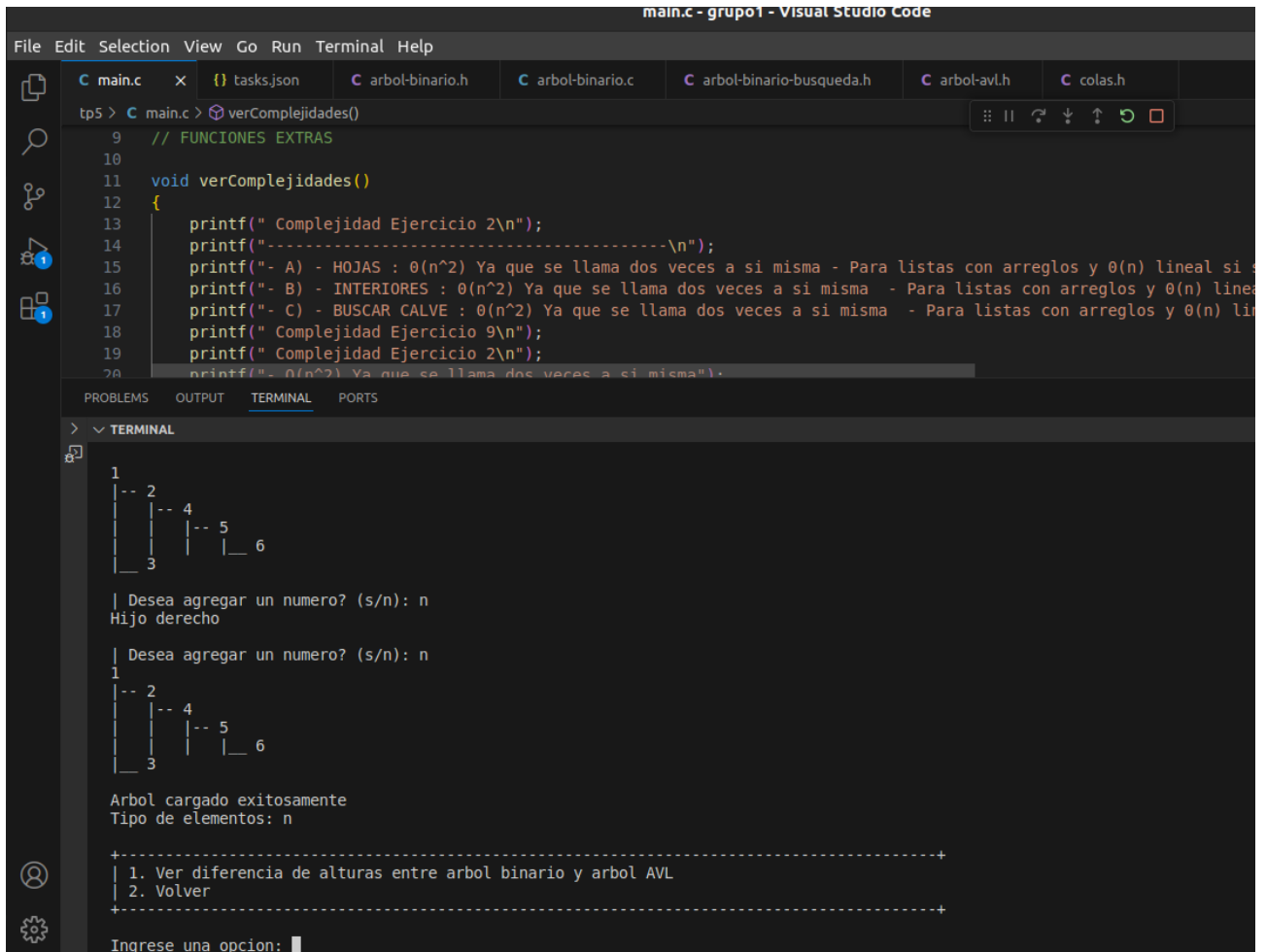
```
tp5 > C main.c > verComplejidades()
9 // FUNCIONES EXTRAS
10
11 void verComplejidades()
12 {
13     printf(" Complejidad Ejercicio 2\n");
14     printf("-----\n");
15     printf("- A) - HOJAS :  $O(n^2)$  Ya que se llama dos veces a si misma - Para listas con arreglo\n");
16     printf("- B) - INTERIORES :  $O(n^2)$  Ya que se llama dos veces a si misma - Para listas con\n");
17     printf("- C) - BUSCAR CALVE :  $O(n^2)$  Ya que se llama dos veces a si misma - Para listas con\n");
18     printf(" Complejidad Ejercicio 9\n");
19     printf(" Complejidad Ejercicio 2\n");
20     printf("-  $O(n^2)$  Ya que se llama dos veces a si misma");
```

The terminal output shows the execution of the program:

```
La altura del arbol es: 2
Presione Enter para continuar...
```

GRUPO 1 – Correcciones Trabajo Práctico: ÁRBOLES

En el ejercicio 9 indica que la diferencia entre los árboles es 2, pero no informa las alturas de los árboles. La complejidad informada es incorrecta.



The screenshot shows the Visual Studio Code editor with a C program named `main.c` open. The program is titled `main.c - grupo1 - Visual Studio Code`. The code defines a function `verComplejidades()` which prints information about complexity for different exercises. The terminal output shows the execution of the program, displaying a binary tree structure and a menu of options.

```
tp5 > C main.c > verComplejidades()
9 // FUNCIONES EXTRAS
10
11 void verComplejidades()
12 {
13     printf(" Complejidad Ejercicio 2\n");
14     printf("-----\n");
15     printf("- A) - HOJAS : 0(n^2) Ya que se llama dos veces a si misma - Para listas con arreglos y 0(n) lineal si s
16     printf("- B) - INTERIORES : 0(n^2) Ya que se llama dos veces a si misma - Para listas con arreglos y 0(n) linea
17     printf("- C) - BUSCAR CALVE : 0(n^2) Ya que se llama dos veces a si misma - Para listas con arreglos y 0(n) li
18     printf(" Complejidad Ejercicio 9\n");
19     printf(" Complejidad Ejercicio 2\n");
20     printf("- 0(n^2) Ya que se llama dos veces a si misma");
```

Terminal Output:

```
1
|-- 2
|  |-- 4
|  |  |-- 5
|  |  |__ 6
|  |__ 3
|__ 3

| Desea agregar un numero? (s/n): n
Hijo derecho

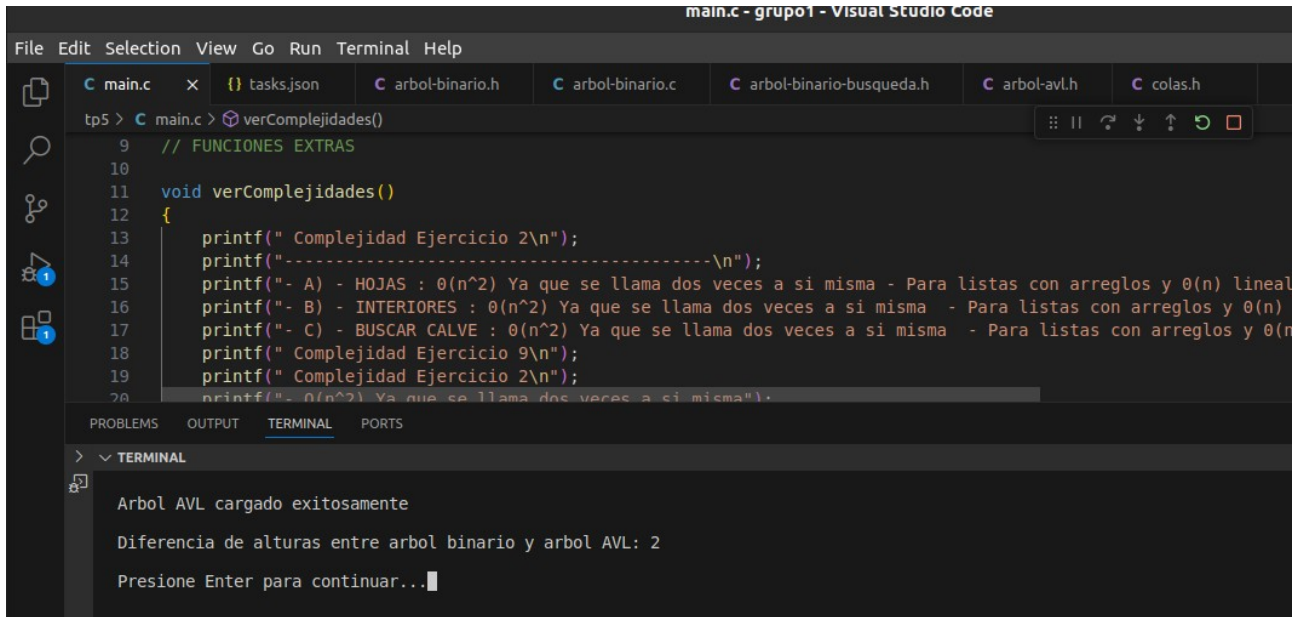
| Desea agregar un numero? (s/n): n
1
|-- 2
|  |-- 4
|  |  |-- 5
|  |  |__ 6
|  |__ 3
|__ 3

Arbol cargado exitosamente
Tipo de elementos: n

+-----+
| 1. Ver diferencia de alturas entre arbol binario y arbol AVL
| 2. Volver
+-----+

Ingrese una opcion: 
```

GRUPO 1 – Correcciones Trabajo Práctico: ÁRBOLES



The screenshot shows the Visual Studio Code editor with the file `main.c` open. The code defines a function `verComplejidades()` that prints complexity information for three exercises. The terminal output shows the program has run successfully, displaying the height difference between a binary tree and an AVL tree as 2.

```
main.c - grupo1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
tp5 > C main.c > verComplejidades()
9 // FUNCIONES EXTRAS
10
11 void verComplejidades()
12 {
13     printf(" Complejidad Ejercicio 2\n");
14     printf("-----\n");
15     printf("- A) - HOJAS : 0(n^2) Ya que se llama dos veces a si misma - Para listas con arreglos y 0(n) lineal\n");
16     printf("- B) - INTERIORES : 0(n^2) Ya que se llama dos veces a si misma - Para listas con arreglos y 0(n)\n");
17     printf("- C) - BUSCAR CALVE : 0(n^2) Ya que se llama dos veces a si misma - Para listas con arreglos y 0(n)\n");
18     printf(" Complejidad Ejercicio 9\n");
19     printf(" Complejidad Ejercicio 2\n");
20     printf("- 0(n^2) Ya que se llama dos veces a si misma\n");
}

PROBLEMS OUTPUT TERMINAL PORTS
> TERMINAL
Arbol AVL cargado exitosamente
Diferencia de alturas entre arbol binario y arbol AVL: 2
Presione Enter para continuar...
```

En el ejercicio 10 la limpieza de pantalla es específica para linux. No validan la cantidad de claves a generar versus el rango de generación de valores al azar. No validan que la cantidad de valores a generar o la cantidad de repeticiones excedan el tamaño máximo de la lista. La función altura retorna uno menos que el valor correcto. Sólo muestran las diferencias de alturas. Faltan las conclusiones.