

LAB 4

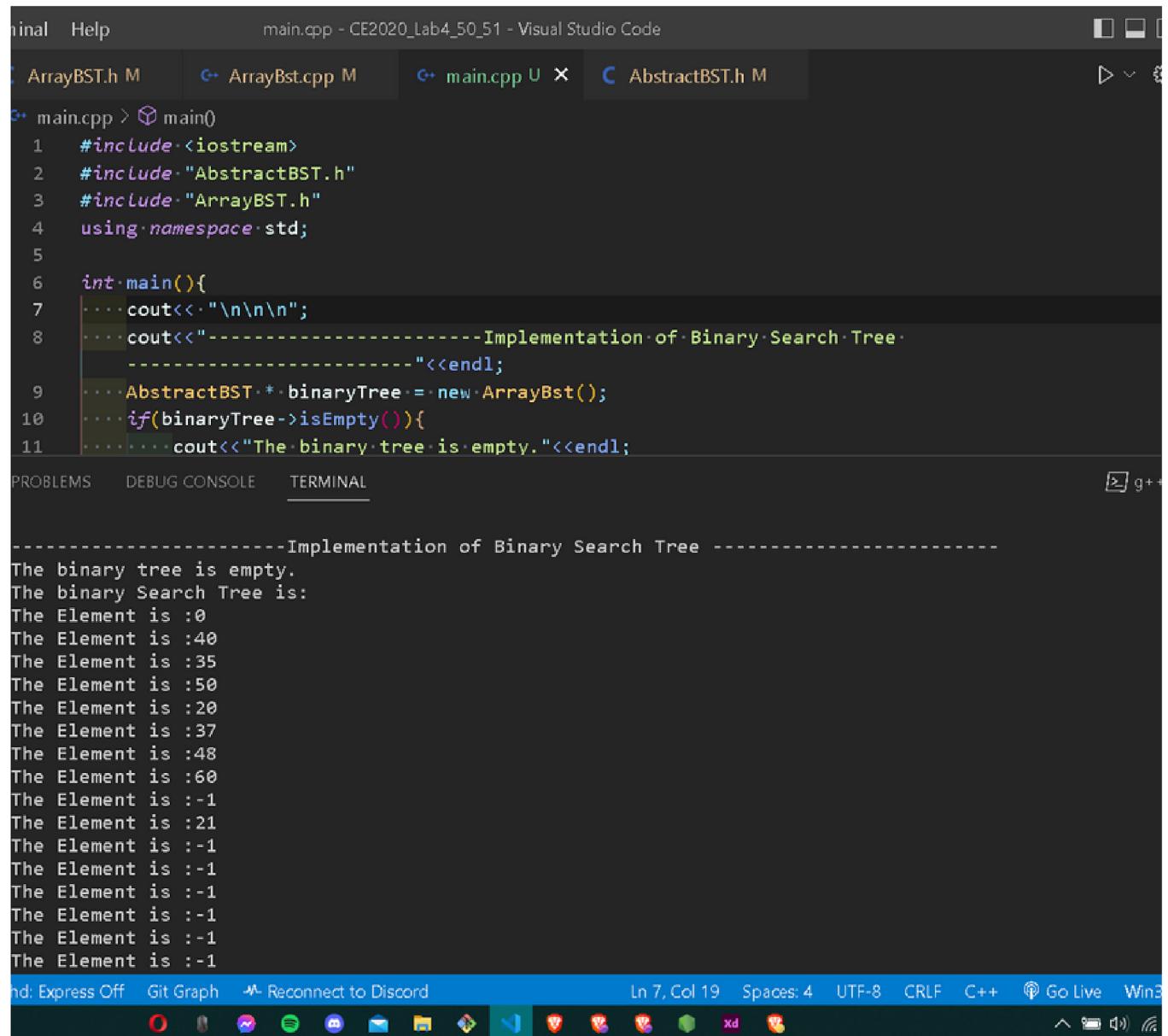
Implementaion of A BST

1.isEmpty()

The function returns true if the search Tree is empty.

With ARRAY:

The function checks if the element at the first position is -1 (considering -1 as a null value). IF so the function returns true.



```
final Help main.cpp - CE2020_Lab4_50_51 - Visual Studio Code
ArrayBST.h M ArrayBst.cpp M main.cpp U X AbstractBST.h M
main.cpp > main()
1 #include <iostream>
2 #include "AbstractBST.h"
3 #include "ArrayBST.h"
4 using namespace std;
5
6 int main(){
7     cout<<"\n\n\n";
8     cout<<"----- Implementation of Binary Search Tree
-----" << endl;
9     AbstractBST *binaryTree = new ArrayBst();
10    if(binaryTree->isEmpty()){
11        cout<<"The binary tree is empty." << endl;
}
PROBLEMS DEBUG CONSOLE TERMINAL g++
```

```
-----Implementation of Binary Search Tree -----
The binary tree is empty.
The binary Search Tree is:
The Element is :0
The Element is :40
The Element is :35
The Element is :50
The Element is :20
The Element is :37
The Element is :48
The Element is :60
The Element is : -1
The Element is :21
The Element is : -1
```

File: Express Off Git Graph Reconnect to Discord Line 7, Col 19 Spaces: 4 UTF-8 CRLF C++ Go Live Win3

2.Add(key,value)

The function adds an element on the correct position in the binary search tree.

At first the function checks if the tree is empty .A while loop runs until the key is -1: If so the new node is added as the root of the tree. If the tree is not empty, the key of the node ToBeAdded is compared with the key of the current root.

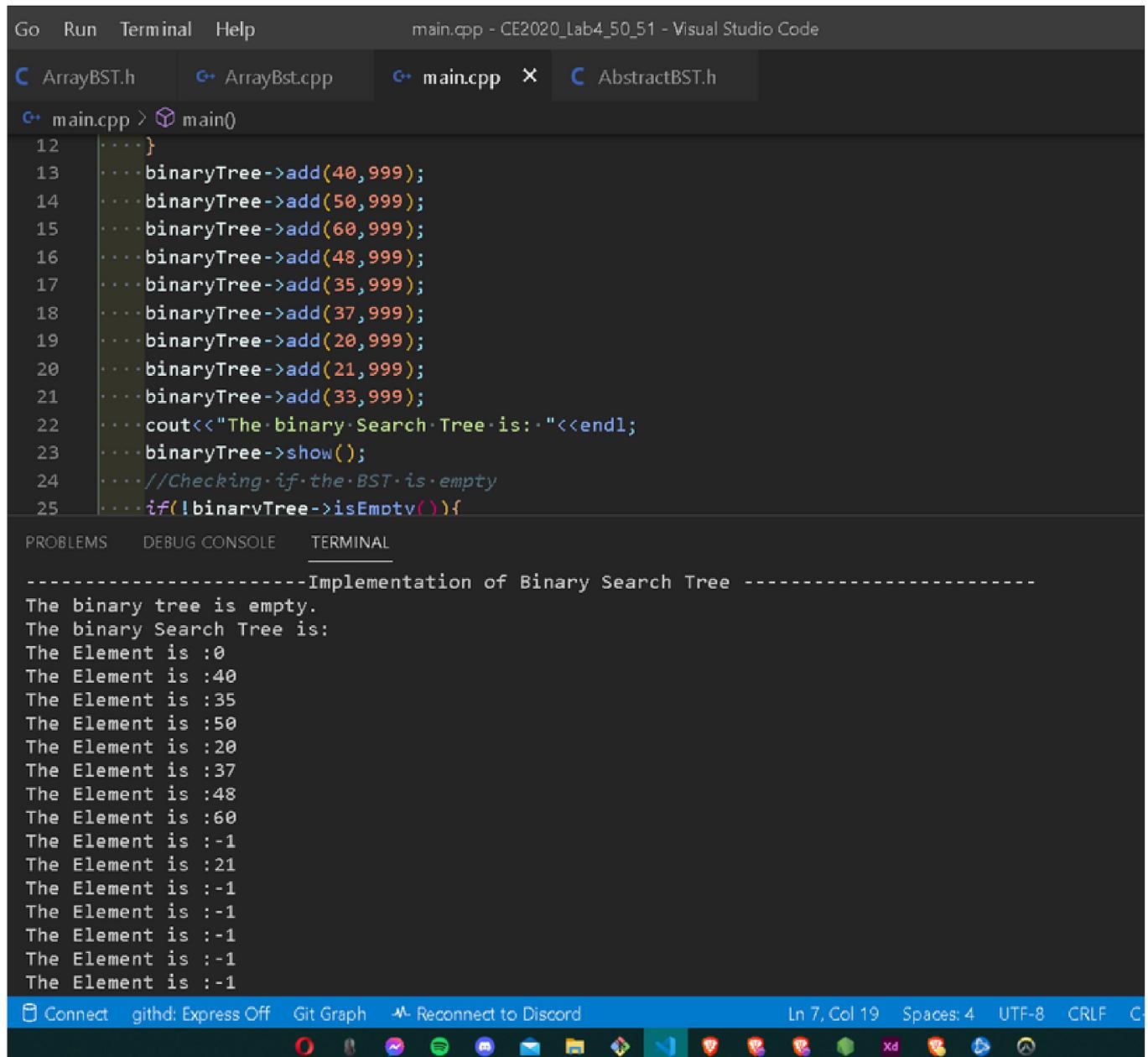
if root.key > nodeToBeAdded.key :

New root is set to be the left child:

else:

new root is set to be the right child:

This process is iterated until an empty node is found.



The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files: ArrayBST.h, ArrayBst.cpp, main.cpp (active), and AbstractBST.h.
- Code Editor:** Displays the `main.cpp` file with code for adding elements to a binary search tree and printing its structure.
- Output:** Shows the terminal output of the program execution.
- Bottom Status Bar:** Includes icons for GitHub, Express, Git Graph, Reconnect to Discord, and various system status indicators.

Code in main.cpp:

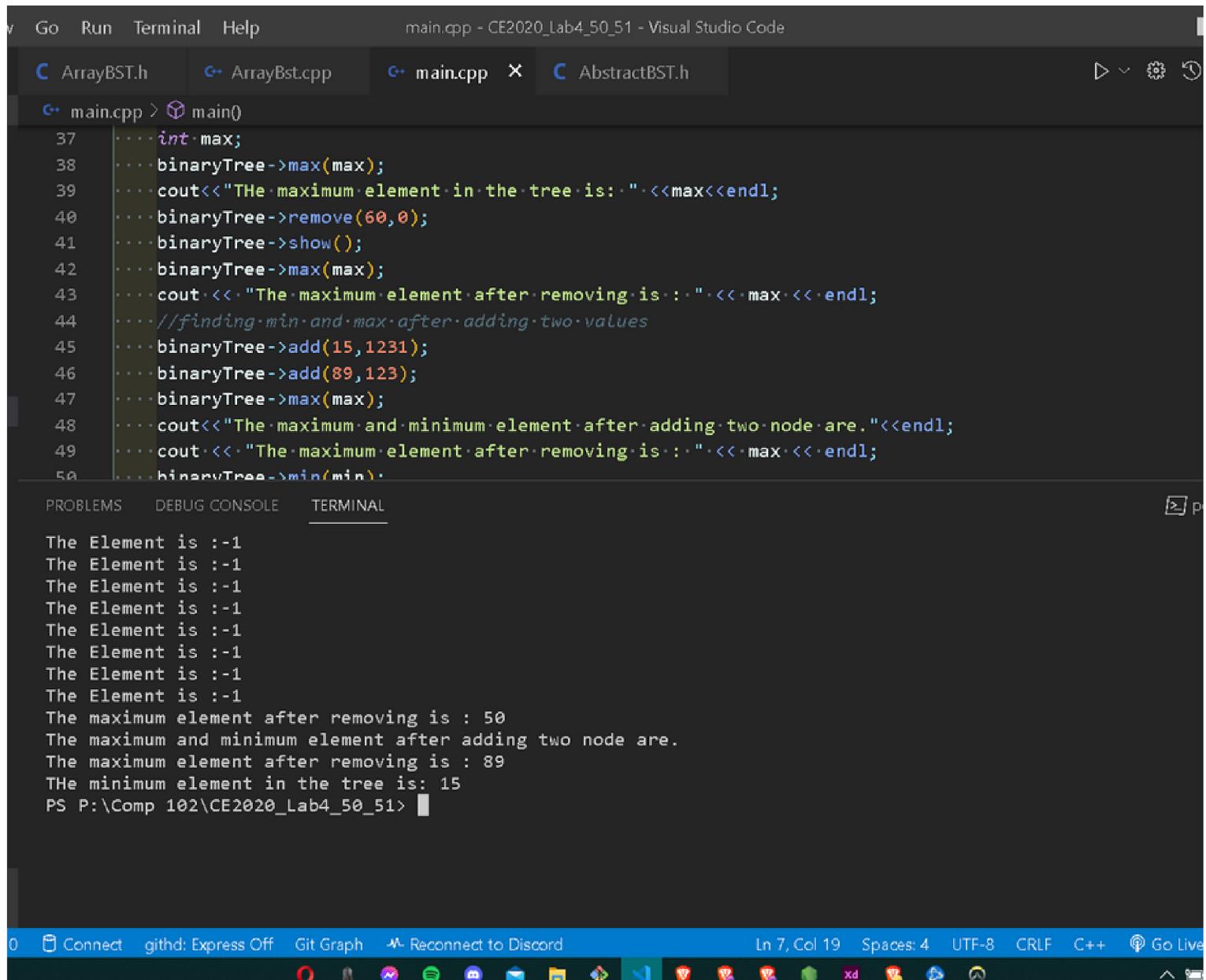
```
12     ....}
13     ....binaryTree->add(40,999);
14     ....binaryTree->add(50,999);
15     ....binaryTree->add(60,999);
16     ....binaryTree->add(48,999);
17     ....binaryTree->add(35,999);
18     ....binaryTree->add(37,999);
19     ....binaryTree->add(20,999);
20     ....binaryTree->add(21,999);
21     ....binaryTree->add(33,999);
22     ....cout<<"The binary Search Tree is::"<<endl;
23     ....binaryTree->show();
24     ....//Checking if the BST is empty
25     ....if(!binaryTree->isEmptyv()){
```

Terminal Output:

```
-----Implementation of Binary Search Tree -----
The binary tree is empty.
The binary Search Tree is:
The Element is :0
The Element is :40
The Element is :35
The Element is :50
The Element is :20
The Element is :37
The Element is :48
The Element is :60
The Element is : -1
The Element is :21
The Element is : -1
```

3.max()

A property of a binary search tree states that the element greater than the node is kept as the right child of the parent node. So the leftmost node is the maximum element in the binary search tree.
A for loop runs until the left most element is figured out.



The screenshot shows the Visual Studio Code interface with the following details:

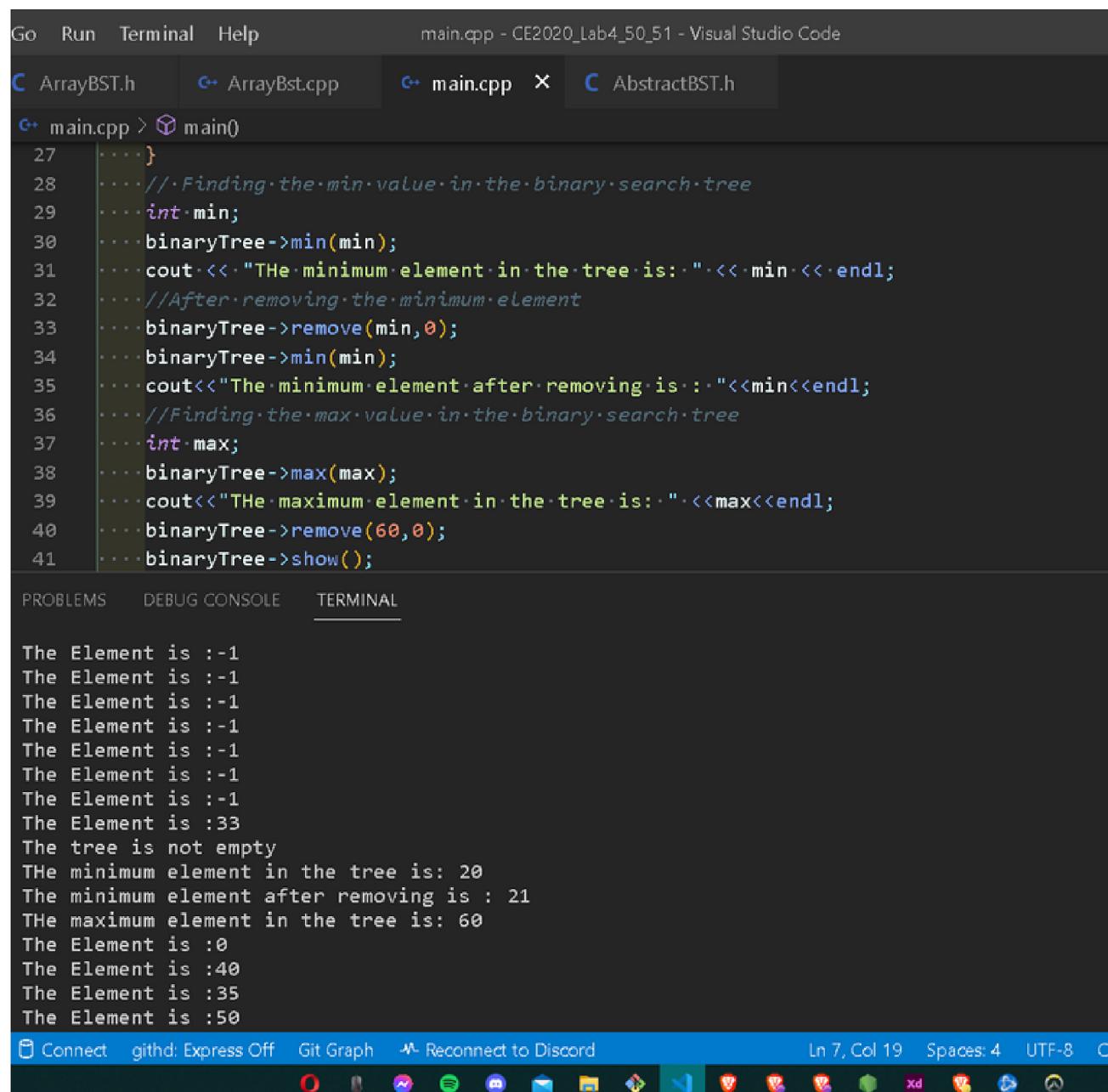
- File Explorer:** Shows files: ArrayBST.h, ArrayBst.cpp, main.cpp (active), and AbstractBST.h.
- Code Editor:** The main.cpp file contains C++ code for a binary search tree. It includes operations for adding nodes, removing nodes, and finding the maximum and minimum elements. The code is as follows:

```
37 |     int max;
38 |     binaryTree->max(max);
39 |     cout<<"The maximum element in the tree is: "<<max<<endl;
40 |     binaryTree->remove(60,0);
41 |     binaryTree->show();
42 |     binaryTree->max(max);
43 |     cout<<"The maximum element after removing is: "<<max<<endl;
44 |     //finding min and max after adding two values
45 |     binaryTree->add(15,1231);
46 |     binaryTree->add(89,123);
47 |     binaryTree->max(max);
48 |     cout<<"The maximum and minimum element after adding two node are."<<endl;
49 |     cout<<"The maximum element after removing is: "<<max<<endl;
50 |     binaryTree->min(min);
```

- Terminal:** Displays the output of the program. The output shows the maximum element being printed multiple times as -1, followed by the maximum element after removing 60 (50), the maximum and minimum elements after adding two nodes (15 and 89), and the maximum element after removing 60 again (89). The minimum element in the tree is also printed as 15.
- Status Bar:** Shows the file path as PS P:\Comp 102\CE2020_Lab4_50_51>, line 7, column 19, spaces: 4, encoding: UTF-8, CRLF, C++, and Go Live.

4.min()

A property of a binary search tree states that the element smaller than the node is kept as the left child of the parent node. So the rightmost node is the minimum element in the binary search tree. A for loop runs until the rightmost element is figured out.



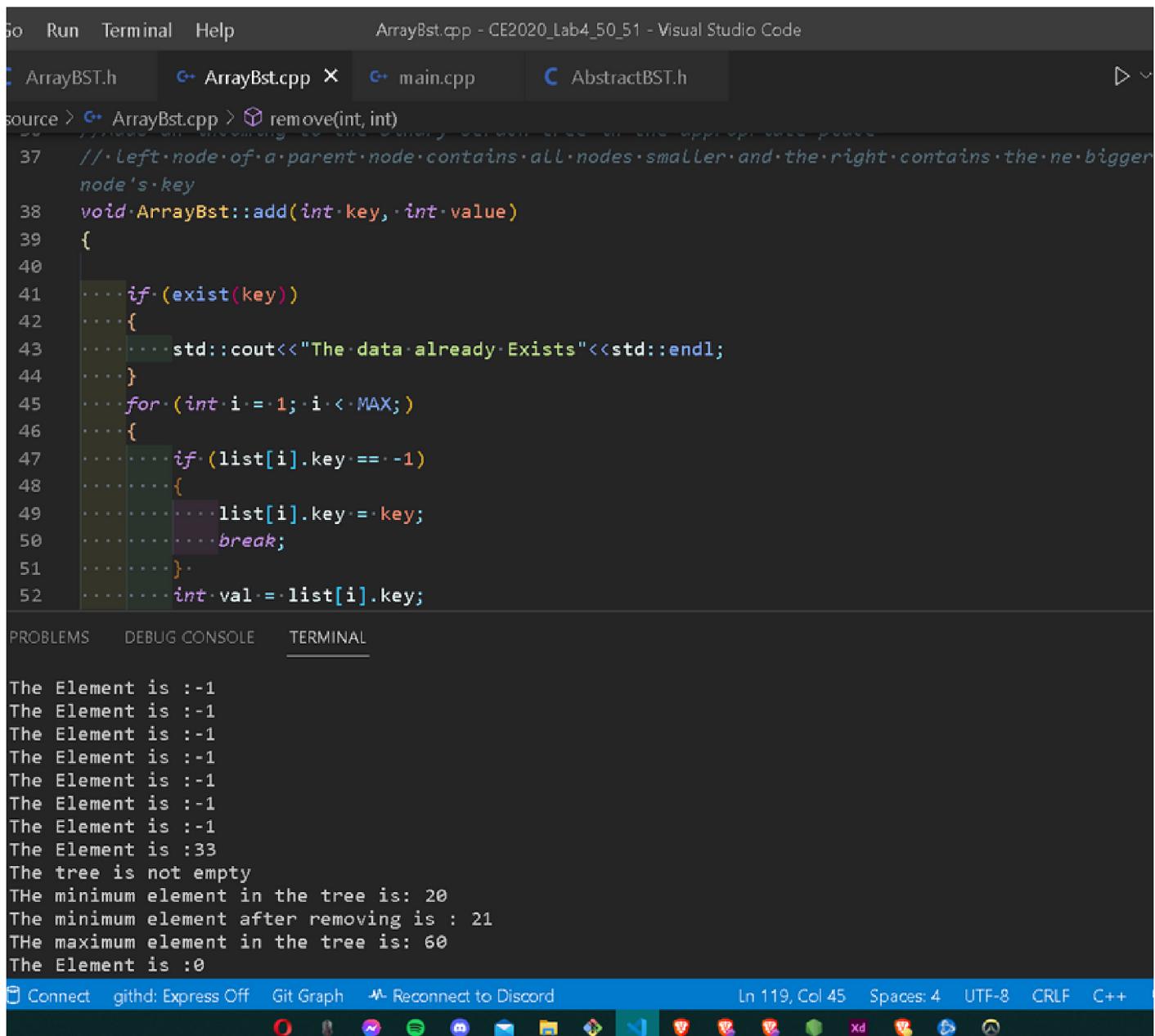
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files: ArrayBST.h, ArrayBst.cpp, main.cpp (selected), and AbstractBST.h.
- Code Editor:** Displays the main.cpp file content. The code implements a binary search tree and demonstrates finding the minimum and maximum values.
- Terminal:** Shows the output of the program execution. The output is:

```
The Element is :-1
The Element is :33
The tree is not empty
THe minimum element in the tree is: 20
The minimum element after removing is : 21
THe maximum element in the tree is: 60
The Element is :0
The Element is :40
The Element is :35
The Element is :50
```
- Bottom Status Bar:** Shows connection status (Connect, githd: Express Off, Git Graph, Reconnect to Discord), file information (Ln 7, Col 19, Spaces: 4, UTF-8), and a toolbar with various icons.

5.exist()

The exist function works similar to that of add. Only exceptions is that the key of node is also checked if it is equal to that of the root. If so true is returned , else iterates until a null node is found and false is returned in that case.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files: ArrayBST.h, ArrayBst.cpp (active), main.cpp, AbstractBST.h.
- Code Editor:** Displays the `ArrayBst.cpp` file. The code implements an array-based binary search tree. It includes a check for existing keys and a loop to find the index for a new key. The code is annotated with comments explaining its logic.
- Terminal:** Shows the output of the program's execution. The output includes:
 - Multiple lines of "The Element is :-1" (likely from a loop or a series of invalid inputs).
 - "The tree is not empty"
 - "THe minimum element in the tree is: 20"
 - "The minimum element after removing is : 21"
 - "THe maximum element in the tree is: 60"
 - "The Element is :0"
- Bottom Bar:** Includes icons for Connect, GitHub, Git Graph, Reconnect to Discord, and various extension icons. Status information: Line 119, Column 45, Spaces: 4, UTF-8, CRLF, C++.

6. show()

The function iterates through each element of the array and prints it in the console. The null node is also printed in this case.

Go Run Terminal Help main.cpp - CE2020_Lab4_50_51 - Visual Studio Code

C ArrayBST.h C ArrayBst.cpp C main.cpp X C AbstractBST.h

main.cpp > main()

```
17     ....binaryTree->add(35,999);
18     ....binaryTree->add(37,999);
19     ....binaryTree->add(20,999);
20     ....binaryTree->add(21,999);
21     ....binaryTree->add(33,999);
22     ....cout<<"The binary Search Tree is:"<<endl;
23     ....binaryTree->show();
24     ....//Checking if the BST is empty
25     ....if(!binaryTree->isEmpty()){
26     ....    cout<<"The tree is not empty"<<endl;
27     ....}
28     ....//Finding the min value in the binary search tree
```

PROBLEMS DEBUG CONSOLE TERMINAL

```
The Element is :40
The Element is :35
The Element is :50
The Element is :20
The Element is :37
The Element is :48
The Element is :60
The Element is :1
The Element is :21
The Element is :1
The Element is :33
```

Connect githd: Express Off Git Graph Reconnect to Discord Ln 7, Col 19 Spaces: 4 UTF-8 CRLF C+

7.remove()

This function works similar to the exist function in the aspect of finding the node to be deleted.

After the node is found , node can have :

i) One child:

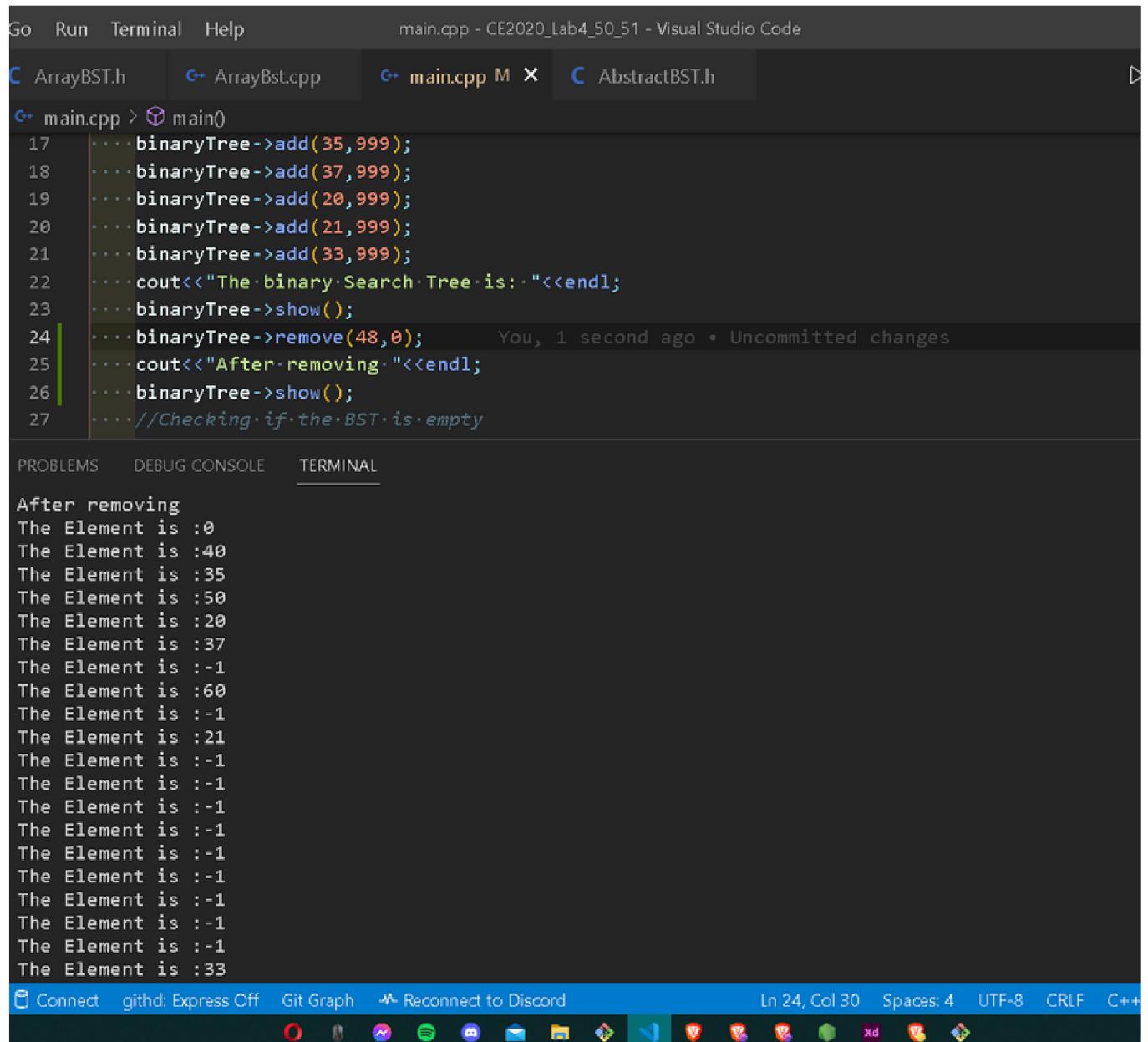
In this case , the node is removed and the child is kept as the new node in the parent's place. . And a function call is made to delete child.

ii) Two children.

In this case the largest element from the left sub tree replaces the node to be deleted. And a function call is made to delete the node with the largest element.

iii)No child:

The node is just deleted.



Go Run Terminal Help main.cpp - CE2020_Lab4_50_51 - Visual Studio Code

C ArrayBST.h C ArrayBst.cpp C main.cpp M X C AbstractBST.h

C main.cpp > main()

```
17     ...binaryTree->add(35,999);
18     ...binaryTree->add(37,999);
19     ...binaryTree->add(20,999);
20     ...binaryTree->add(21,999);
21     ...binaryTree->add(33,999);
22     ...cout<<"The binary Search Tree is: "<<endl;
23     ...binaryTree->show();
24     ...binaryTree->remove(48,0);      You, 1 second ago • Uncommitted changes
25     ...cout<<"After removing: "<<endl;
26     ...binaryTree->show();
27     ...//Checking if the BST is empty
```

PROBLEMS DEBUG CONSOLE TERMINAL

After removing

```
The Element is :0
The Element is :40
The Element is :35
The Element is :50
The Element is :20
The Element is :37
The Element is :-1
The Element is :60
The Element is : -1
The Element is :21
The Element is : -1
The Element is :33
```

Connect githd: Express Off Git Graph Reconnect to Discord Ln 24, Col 30 Spaces: 4 UTF-8 CRLF C++