Assignment 9.1	
Implementing Trees	
Course Code: CPE010	**Program:** BSCPE
Course Title: Database Management System	**Date Performed:** Sep 25, 2025
Section: CPE21S4	**Date Submitted:** Sep 25, 2025
Name: Bautista,Mariela S.	**Instructor:** Engr. Quejado

1. Objective(s):

Objectives: To create a program in C++ using search trees.

1. To be familiarized with the search tree algorithm.

2. To be able to create a program with search tree algorithm.

Output:

Answer the following questions:

1. Define what is a binary tree?

A **binary tree** is a hierarchical or ranked data structure where each node has at most two children referred to as the left and right child. This is commonly used in computer science because of storage efficiency and, extracting and specific information from databases and data storage systems. Which also has a various operations such as insertion,deletion, and traversal.

2. What are the key components of a Binary tree?

The key components of a Binary Tree are **Nodes** and the link between them. A node is the basic building block of the tree, each node usually stores a **data value**. Along with the data, every node also has two pointers or references: one for the left child and one for the right child. The **root node** is the very first node at the top of the tree, and its where the structure begins. The left child and right child can each be either another node or NULL(if there is no child in that direction). Together, tehse components allow the binary tree to grow, branch out, and store data in a structured way.

3. What are the operations that can be performed in a Binary tree?

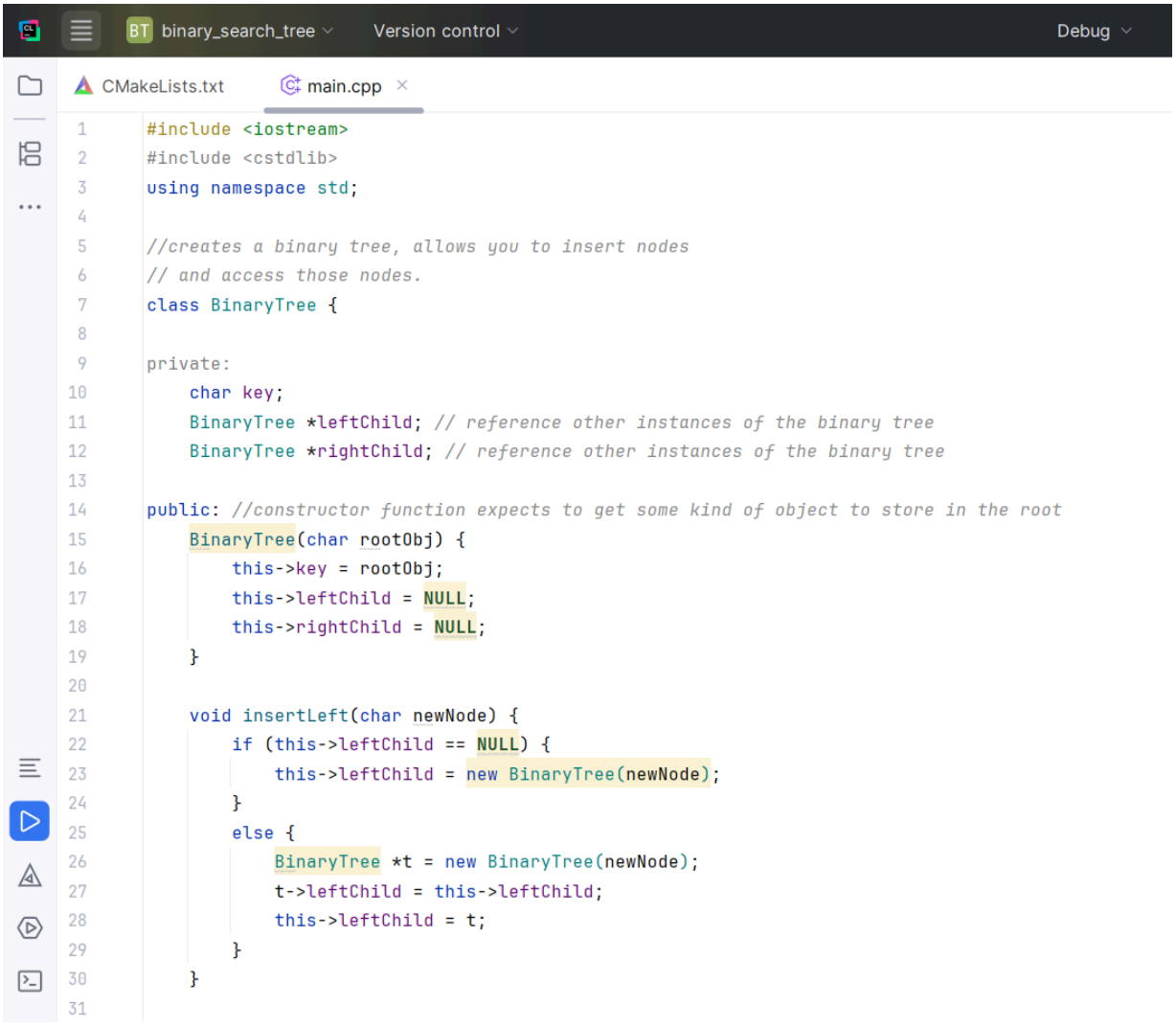
The operations that can be performed in a binary tree, and this are: We can **insert** a new node into the tree. We can **delete** an existing node and restructure the tree accordingly. **Traversal** operations like

inorder, preorder, and postorder are used to visit nodes. **Searching** for a value is one of the most common operations. And also, **Balancing** operations can also be applied to optimize performance.

4. What are the conditions for a Binary Search Tree?

The conditions of the Binary Tree are: **Node Structure** wherein each node in a binary tree consists of three components which are; data, a pointer to the left child, and a pointer to the right child. Where this allows for a maximum of two children each node. **Height and Depth** wherein the height of the binary tree is defined as the number of edges from the root to that node. **Maximum Nodes** wherein a binary tree of height h

5. Give a simple sample program in C++ that uses the above algorithm. Explain how the program works.




```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

//creates a binary tree, allows you to insert nodes
// and access those nodes.
class BinaryTree {

private:
    char key;
    BinaryTree *leftChild; // reference other instances of the binary tree
    BinaryTree *rightChild; // reference other instances of the binary tree

public: //constructor function expects to get some kind of object to store in the root
    BinaryTree(char rootObj) {
        this->key = rootObj;
        this->leftChild = NULL;
        this->rightChild = NULL;
    }

    void insertLeft(char newNode) {
        if (this->leftChild == NULL) {
            this->leftChild = new BinaryTree(newNode);
        }
        else {
            BinaryTree *t = new BinaryTree(newNode);
            t->leftChild = this->leftChild;
            this->leftChild = t;
        }
    }

```

```
31
32      void insertRight(char newNode) {
33          if (this->rightChild == NULL) {
34              this->rightChild = new BinaryTree(newNode);
35          }
36          else {
37              BinaryTree *t = new BinaryTree(newNode);
38              t->rightChild = this->rightChild;
39              this->rightChild = t;
40          }
41      }
42
43      BinaryTree *getRightChild() {
44          return this->rightChild;
45      }
46
47      BinaryTree *getLeftChild() {
48          return this->leftChild;
49      }
50
51      void setRootVal(char obj) {
52          this->key = obj;
53      }
54
55      char getRootVal() {
56          return this->key;
57      }
58  };
59
```

```cpp
int main() {
    BinaryTree *r = new BinaryTree( rootObj: 'a');
    cout << r->getRootVal() << endl;

    cout << r->getLeftChild() << endl;
    r->insertLeft( newNode: 'b');
    cout << r->getLeftChild() << endl;
    cout << r->getLeftChild()->getRootVal() << endl;

    r->insertRight( newNode: 'c');
    cout << r->getRightChild() << endl;
    cout << r->getRightChild()->getRootVal() << endl;

    // changing the key on the right node of the tree
    r->getRightChild()->setRootVal( obj: 'd');
    cout << r->getRightChild()->getRootVal() << endl;

    // inserting e
    r->getLeftChild()->insertLeft( newNode: 'e');
    cout << r->getLeftChild()->getLeftChild()->getRootVal() << endl;

    // inserting f
    r->getLeftChild()->insertRight( newNode: 'f');
    cout << r->getLeftChild()->getRightChild()->getRootVal() << endl;

    // inserting g
    r->getLeftChild()->getLeftChild()->insertLeft( newNode: 'g');
    cout << r->getLeftChild()->getLeftChild()->getLeftChild()->getRootVal() << endl;

    return 0;
}
```

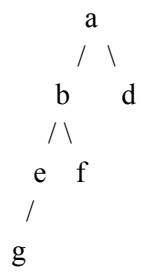
Output:

```
Run    binary_search_tree

C:\Users\Mariela\CLionProjects\binary_search_tree\cmake-build-debug\binary_search_tree.exe
a
0
0x1594cfb1b10
b
0x1594cfb1b30
c
d
e
f
g

Process finished with exit code 0
```

Conclusion:

This code creates a **binary tree**, which means every node can have a left child and right child. Firstly, what i did is i make the root node with the value ‘a’ and printed it out. At the beginning the left child is empty “NULL”, but when I insert ‘b’, it beocmes the left child of ‘a’ and i can see its value when i print it. After that, i insert ‘c’ as the right child of ‘a’, but then i change it to ‘d’ using the setRootVal function, which shows that i can update the value of a node. Next, i add ‘e’ as the left of ‘b’, and then i insert ‘f’ as the right child of ‘b’, so now ‘b’ has two children. After that, i insert ‘g’ under ‘e’, making the tree go one level deeper on the left side.

And now, my tree will look like this:



Each time i use cout, it shows me either the value of a node or the memory address of a child pointer. The most important outputs are the characters (a, b, c, d, e, f, g) which prove that the insertion and changes worked. This helped me see how the binary tree grows step by step and how i can both insert new nodes and update existing nodes.

References:

GeeksforGeeks. (2025, August 2). *Binary tree data structure*. https://www.geeksforgeeks.org/dsa/binary-tree-data-structure/

GeeksforGeeks. (2025, August 2). *What is binary tree?*

https://www.geeksforgeeks.org/dsa/what-is-binary-tree/

Tip (n.d.). *9.1 Trees*. In *Course pages*. Retrieved from https://tip.instructure.com/courses/66762/pages/9-dot-1-trees?module_item_id=7567590