Quiz No. <9.1>	
<Tree Structure>	
Course Code: CPE010	**Program: BSCPE**
Course Title: Data Structures and Algorithms	**Date Performed: 10/2/2025**
Section: CPE21S4	**Date Submitted: 10/2/2025**
Name(s): Anastacio, Lester Arvid P.	**Instructor: Engr. Jimlord Quejado**

6. Output

7. Supplementary Activity

I. **Define the following**

A. Tree

- A Tree structure in data structures is that it is an algorithm that is hierarchical data structure that holds the nodes, this consists of nodes wherein one holds a value which is linked to a child node, it starts from it being a single point wherein it is the root and it branches out as more data is being stored within it, a similar analogy to this is the family tree in which we have the parents of our parents alongside their siblings and so on, which means that a tree is just a way to arranged bits of data to their corresponding parents and data's that are alike.

B. Child node

- This node descended from another node in which from their parent node, these are the little bits of data that are connected just below their parent nodes in which these nodes can also have child nodes of their owns if it is expanded further similar to a family tree for instance in which a family have a father or mother and their own child in which that said child will grow and soon enough will have created a new branch or family of their own.

C. Parent node

- This is the node that more than one or more child nodes connected to it, this is where the child nodes are connected to as this is the very backbone of the tree data structure as this is the node in which arranges the child nodes or in some words the guide or path for data to spread efficiently to the ever growing structure, in which it helps out data be distributed to the suitable nodes.

D. Root node

- This is the start or the topmost of the tree structure, this is where the whole tree begins in which it has no parent nodes but this is where the nodes descended from, if I where to compare this, it is like a business, wherein the owner the creator of a rising business has become quite successful and grew his/her business further, in which created branches or departments to different other places to further spread his business reach even further.

E. Tree Traversal

- The Tree Traversal is the process of visiting all the nodes within the tree data structure in which moves in a specified order it's like taxi in which has different ways to bring in their clients to their designated spots and have their own routes they follow but still have the same path towards the correct designation of their client.

F. Parse Tree

- A Parse Tree is a type of tree structure that represents the syntactic structure of a string according to a formal grammar I think this is like a type of data gathering basing on their grammar like it is sorted out in an order that it should follow.

G. Order of Precedence

- This the rule in which define the sequence in which operations are performed in expressions its like a model wherein you type in symbols which represents the addition, subtraction and other operations, I think this is more of a function that reads and perform the inputted operations basically something that is used in calculations.

H. Parsing

- This is the process of analyzing a string of symbols to determine its grammatical structure with respect to a given formal grammar so basically similar to the Parse tree but this is more of the main function that you could see within the Parse Tree structure, as this one analyzes the process of the strings and symbols to determine whether it is grammatically structured and it is within close to the formal structure, its like a grammar checker to ensure if you type it in correctly or followed the correct rules.

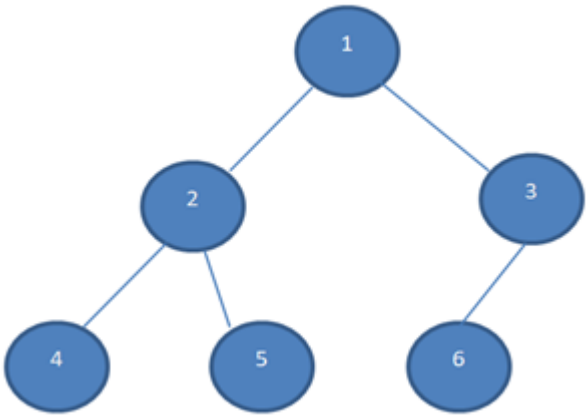
I. Preorder Traversal

- A traversal method where the current node is visited first, followed by its left subtree, then its right subtree, it means that it will visit the parent first before moving on to the left subtree and afterwards moving on to the right tree.

J. Inorder Traversal

- A traversal method where the left subtree is visited first, then the current node, and finally the right subtree.

II. Identify the parts of the following tree



1. Root Node
2. Left Subtree/Parent Node
3. Right Subtree/Parent Node
4. Child Node
5. Child Node
6. Child Node

III. Delete the Node with an element value of 2 in the above tree structure.

PSEUDOCODE:

// Added in class BinaryTree

Function deleteNode(target):
Initialize an empty queue
Enqueue the root node

Set targetNode to NULL
Set lastNode to NULL
Set parentOfLast to NULL

While the queue is not empty:
Dequeue a node and assign it to temp

If temp's value equals target:
Set targetNode to temp

If temp has a left child:
Set parentOfLast to temp
Enqueue temp's left child

If temp has a right child:
Set parentOfLast to temp
Enqueue temp's right child

Set lastNode to temp

If targetNode and lastNode are not NULL:
Replace targetNode's value with lastNode's value

If parentOfLast's right child is lastNode:
Delete parentOfLast's right child
Set parentOfLast's right child to NULL

Else if parentOfLast's left child is lastNode:
Delete parentOfLast's left child
Set parentOfLast's left child to NULL

// In int main()
Call deleteNode with target value '2' on the root node
Print the value of the left child of the root node with label "Node Deleted"

IV. Create a program in C++ to create a binary tree that looks like in the above figure.

CODE:

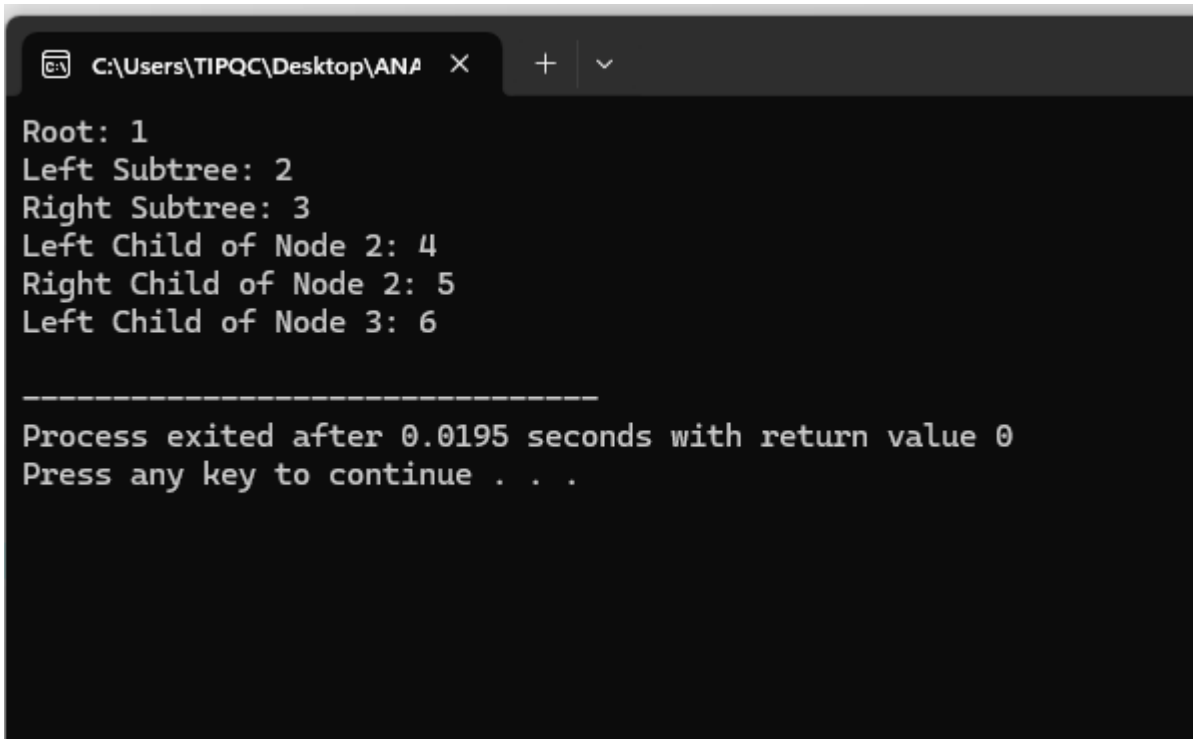
rees.cpp

```cpp
#include <iostream>
#include <cstdlib>

class BinaryTree {
private:
    char key;
    BinaryTree *leftChild;
    BinaryTree *rightChild;

public:

    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;
        }
    }

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

```

Trees.cpp

```cpp
    BinaryTree *getRightChild() {
        return this->rightChild;
    }

    BinaryTree *getLeftChild() {
        return this->leftChild;
    }

    void setRootVal(char obj) {
        this->key = obj;
    }

    char getRootVal() {
        return this->key;
    }
};

int main() {
    BinaryTree *r = new BinaryTree('1');

    r->insertLeft('2');
    r->insertRight('3');

    r->getLeftChild()->insertLeft('4');
    r->getLeftChild()->insertRight('5');

    r->getRightChild()->insertLeft('6');

    std::cout << "Root: " << r->getRootVal() << std::endl;
    std::cout << "Left Subtree: " << r->getLeftChild()->getRootVal() << std::endl;
    std::cout << "Right Subtree: " << r->getRightChild()->getRootVal() << std::endl;
    std::cout << "Left Child of Node 2: " << r->getLeftChild()->getLeftChild()->getRootVal() << std::endl;
    std::cout << "Right Child of Node 2: " << r->getLeftChild()->getRightChild()->getRootVal() << std::endl;
    std::cout << "Left Child of Node 3: " << r->getRightChild()->getLeftChild()->getRootVal() << std::endl;

    return 0;
}
```

OUTPUT:



8. Conclusion

In this Quiz about the Tree structure, it was a very challenging endeavor as I try to define each of these parts of the tree data structure and how each functions and correspond to one another, I even use analogies to better capture the essence of what that part actually do from comparing it to family trees to business and taxi, I also got confused at the third part of the quiz as I don't seems to understand if I should create a code to delete the node 2 or to just simply delete

it within the given image, which is why I just went with it and added a delete function within the code meant for the part IV, and overall, this quiz actually managed to even further enhanced the way of my understand towards the parts of the tree data structure as it shows how I used different analogies to tackle the definitions of the functions.
9. Assessment Rubric