Activity No. 9.2	
Implementing Trees	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: Sept. 25, 2025**
Section: CPE21S4	**Date Submitted: Sept. 25, 2025**
Name(s): Crishen Luper S. Pulgado	**Instructor: Sir Jimlord Quejado**

1. What is a binary search tree?

A binary search tree is a type of binary data structure wherein the nodes on the left child of the tree are lower values of the root, while the right child has a higher value of the root. Furthermore, nodes should not be duplicated. It is a hierarchal structure where it has subtree which is the left and right keys. We can use functions such as searching, inserting, and deletion of elements without shifting the values in memory.

2. Where can binary search tree be used?

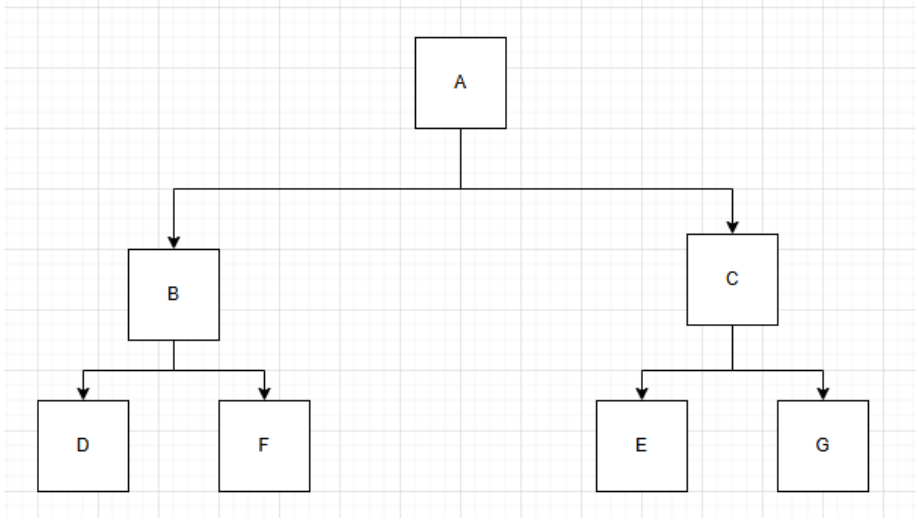
We use self-balancing BST where it will maintain the height as O(Log n). The complexity of the BST is O(h) where h is the height of the BST. So, we can use self-balancing so that it the operations will be O(log n). BST can be used in getting online orders placed while maintaining the live data in sorted in prices. It can also be used in doubly ended priority queue where we can both use max heap and minimum heap. Moreover, it can also be used for large data set via inserting elements of the dataset into a BST.

3. What is a tree traversal.

Tree traversal refers to the process of visiting each node of a tree once in a certain order. Since all nodes are connected through edges or links, we will always start from the root node. Hence, we cannot randomly access a node in a tree. There are different ways which we can traverse a tree, which are In-order, pre-order, and post-order traversals.

4. Differentiate a post-order and pre-order traversal give examples.

Post-order and pre-order traversals are ways to traverse a tree. The difference is that in the post-order, the root is visited first then it will visit the left subtree and lastly the right subtree. So for example:



Here, we will start at the root A. Since the left subtree should be visited first, we will traverse to B then D (A->B->D). Next, is the right child of the B which is F. Since, we have traversed all the left subtree, we will traverse into the right which is C. As always, we should visit the left subtree which is the E then G will be the last. Hence the traversal is A->B->D->F->C->E->G.

In post-order, we will traverse first the left subtree, then the right subtree, and lastly, we will visit the root. Based on the sample, we initially traverse from the room to the leftmost node which is D. So, D will be the first node to be visited since D has no subtree. After D, we will traverse to F and then traverse to B (D->F->B). Next is the right subtree but it will visit next is the left of the C which is E. Then, traverse to G then C ((D->F->B->E->G->C). Then lastly, we will visit to A.

4. What is a parse tree and its use?

A parse tree is a tree structure which represents the structure of a given input according to a formal grammar. It is crucial in checking if the input aligns with the language defined by a grammar. It is derived from the start symbol which serves as

the root of the parse tree. It is like dissecting a sentence which the subtree has to be aligned with the root of it. For example, is in a sentence, there is a noun phrase and a verb (subtree). Under a noun phrase, it has a noun and under the verb phrase, it has a verb. So, these aligns with the sentence which has verb and noun.