Series No. 03 – Trees

Exercise 01: Comparing Trees

The goal of this exercise is to write a function to compare two binary trees. Two trees are considered equal if their structures are identical and if the values stored in the nodes are identical. Two trees are considered different if their structures are different or if the values stored in the nodes are different.

1. Define the data structure for representing a binary tree of integers.
2. Write a recursive function to compare two binary trees.
3. Write the iterative version of the previous function.

Exercise 02: Symmetrical Trees

A symmetric binary tree is a special type of binary tree where the left and right subtrees of its root are mirrors of each other relative to each other. In other words, if you imagine the tree as a geometric structure, the left part of the root is the mirror reflection of the right part.

For this exercise, you need to write a function that determines whether a binary tree is symmetric or not. The function returns ***1 (true)*** if the tree is symmetric and ***0 (false)*** otherwise.

Exercise 03: The Sum of a Given Path

Write a function that takes a binary tree and an integer as parameters, and tests whether there is a path from the root to a leaf of the tree whose sum is equal to the integer passed as parameter. The function should return true if the path exists, and false otherwise.

Exercise 04: Convert to AVL

A balanced binary search tree is a binary search tree whose left and right subtrees have equal height or differ by at most 1 for each node. The goal of this exercise is to convert a sorted array into a balanced binary search tree. Write a function "convertAVL" that takes a sorted array as a parameter and returns a balanced binary search tree constructed from the elements of the array.

The function should not perform any rotation. It should simply construct the tree from the array by inserting the elements in the correct position.

Exercise 05: Correct a Binary Search Tree

Given a binary search tree, the values of two nodes have been swapped by mistake. The goal of this exercise is to find and swap the two values without changing its tree structure or recreating it. You must write a function that takes the root of the tree as input and performs the correction by swapping the values of the incorrectly placed nodes.

Hint: the infix traversal of a binary search tree must return the node values in ascending order.