Binary Tree

Practice Programs - 1

1. Write a program to implement the insertion operation in a binary tree.

   - Note that you need to take the following as input:

     i) the number to be inserted;

     ii) the parent;

     iii) whether right or left child.

2. Write a function `void display(btNode *root)` to display the binary tree in the screen properly.

3. Write a function `void inorder_nonrecursive(btNode *root)` that implements the non-recursive inorder traversal of a given binary tree.

4. Write a function `void postorder_nonrecursive(btNode *root)` that implements the non-recursive postorder traversal of a given binary tree.

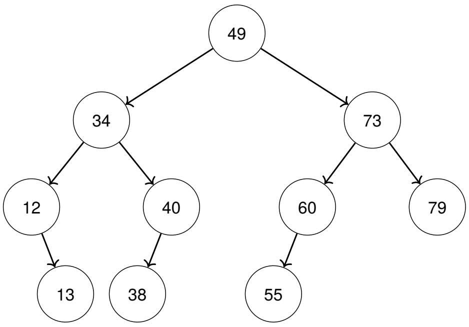
5. Write a function `int findMax(btnode *root)` that returns the maximum number stored in a binary tree.

6. Write a function `int isExists(btNode *root, int d)` that returns `1` if `d` is present in the tree; else returns `0`.

7. Write a function `int getHeight(btNode *root)` that returns the height of a binary tree given the root.

8. Write a function `int getLevel(btNode *root, int d)` that returns the level of `d` in a binary tree given the root.

9. Write a function `void levelorder(btNode *root)` that returns the level order traversal of a binary tree given the root. For example, the level order traversal of the following tree is: 49 34 73 12 40 60 79 13 38 55



10. Write a function `void zigzag(btNode *root)` that returns the level order traversal in a zigzag (alternative traversal in direction from left to right, then from right to left and so on) of a binary tree given the root. For example, the zigzag traversal of the above tree is:
    49 73 34 12 40 60 79 55 38 13

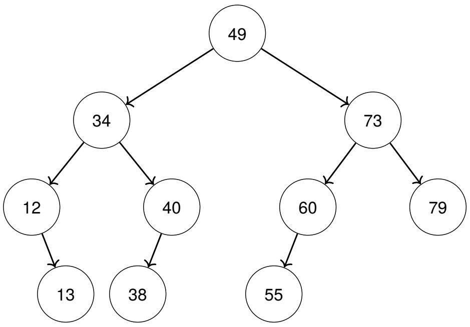
11. Write a function `int getLeafCount(btNode *root)` that returns the number of leaf nodes in a binary tree given the root.

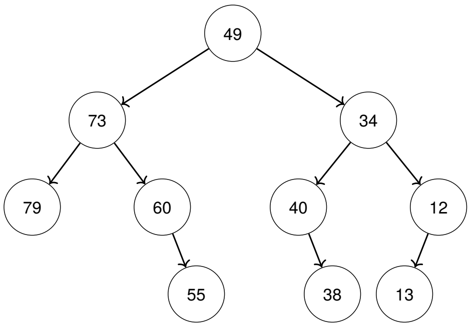
12. The nodes with both left and right children are called **full nodes**. Write a function `int getFullNodeCount(btNode *root)` that returns number of full nodes in a binary tree.

13. The nodes with only one child (either left or right) are called **half nodes**. Write a function `int getHalfNodeCount(btNode *root)` that returns the number of half nodes in a binary tree given the root.

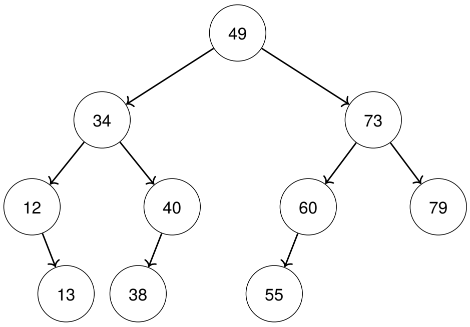
14. Write a function `int isIdentical(btNode *root1, btNode *root2)` that returns `1` if the two binary trees passed as arguments are identical; else returns `0`.

15. Write a function `int isMirror(btNode *root1, btNode *root2)` that returns `1` if the two binary trees passed as arguments are mirror to each other; else returns `0`. An example of mirror binary trees are given below:





16. Write a function `void printAncestors(btNode *root, btNode *node)` that prints all the ancestors of `node`. For example, the anscestors of **40** is **34, 49** for the following tree:



17. Write a function `int getBreadth(btNode *root)` that returns the breadth or the maximum width of the binary tree. For example, the breadth (or width) of the above tree is 4.

18. Write a program that takes positive integers from the user as input and forms a complete binary tree with the input. If the user has given either a negative or a zero input, the program should display the tree and terminates.