In [2]:

```
class Node:
    def __init__(self, key):
        self.data = key
        self.left = None
        self.right = None

# Function to print inorder traversal of tree
def inorderTraversal(root):
    if root:
        inorderTraversal(root.left)  # Visit left subtree
        print(root.data, end=" ")     # Visit root
        inorderTraversal(root.right)  # Visit right subtree

if __name__ == '__main__':
    root = Node(input("Enter root node: "))
    root.left = Node(input("Enter left to root node: "))
    root.right = Node(input("Enter right to root node: "))
    root.right.left = Node(input("Enter left to right of root node: "))
    root.left.left = Node(input("Enter left to left of root node: "))
    root.left.right = Node(input("Enter right to left of root node: "))
    root.left.left.left = Node(input("Enter left to left to left of roo
    root.left.left.right = Node(input("Enter right to left to left of r

    print("Inorder DFS traversal of the binary tree:")
    inorderTraversal(root)

```

```
Enter root node: A
Enter left to root node: B
Enter right to root node: C
Enter left to right of root node: F
Enter left to left of root node: D
Enter right to left of root node: E
Enter left to left to left of root node: G
Enter right to left to left of root node: H
Inorder DFS traversal of the binary tree:
G D H B E A F C
```

In [11]:

```
# Python3 program to for tree traversals

# A class that represents an individual node in a
# Binary Tree


class Node:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.val = key


# A function to do inorder tree traversal
def printInorder(root):

    if root:

        # First recur on left child
        printInorder(root.left)

        # then print the data of node
        print(root.val),

        # now recur on right child
        printInorder(root.right)


# A function to do postorder tree traversal
def printPostorder(root):

    if root:

        # First recur on left child
        printPostorder(root.left)

        # the recur on right child
        printPostorder(root.right)

        # now print the data of node
        print(root.val),


# A function to do preorder tree traversal
def printPreorder(root):

    if root:

        # First print the data of node
        print(root.val),

        # Then recur on left child
        printPreorder(root.left)

        # Finally recur on right child
        printPreorder(root.right)


# Driver code
root = Node(1)
root.left = Node(2)
```

```
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
print ("Preorder traversal of binary tree is")
printPreorder(root)

print ("\nInorder traversal of binary tree is")
printInorder(root)

print("\nPostorder traversal of binary tree is")
printPostorder(root)

```

```
Preorder traversal of binary tree is
1
2
4
5
3

Inorder traversal of binary tree is
4
2
5
1
3

Postorder traversal of binary tree is
4
5
2
3
1
```

In [ ]:

```
1
```