

In [2]:

```

1 class Node:
2     def __init__(self, key):
3         self.data = key
4         self.left = None
5         self.right = None
6
7     # Function to print inorder traversal of tree
8     def inorderTraversal(root):
9         if root:
10             inorderTraversal(root.left) # Visit left subtree
11             print(root.data, end=" ")   # Visit root
12             inorderTraversal(root.right) # Visit right subtree
13
14 if __name__ == '__main__':
15     root = Node(input("Enter root node: "))
16     root.left = Node(input("Enter left to root node: "))
17     root.right = Node(input("Enter right to root node: "))
18     root.right.left = Node(input("Enter left to right of root node: "))
19     root.left.left = Node(input("Enter left to left of root node: "))
20     root.left.right = Node(input("Enter right to left of root node: "))
21     root.left.left.left = Node(input("Enter left to left to left of root node: "))
22     root.left.left.right = Node(input("Enter right to left to left of root node: "))
23
24     print("Inorder DFS traversal of the binary tree:")
25     inorderTraversal(root)
26

```

```

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]:

```
1  # Python3 program to for tree traversals
2
3  # A class that represents an individual node in a
4  # Binary Tree
5
6
7  class Node:
8      def __init__(self, key):
9          self.left = None
10         self.right = None
11         self.val = key
12
13
14 # A function to do inorder tree traversal
15 def printInorder(root):
16
17     if root:
18
19         # First recur on left child
20         printInorder(root.left)
21
22         # then print the data of node
23         print(root.val),
24
25         # now recur on right child
26         printInorder(root.right)
27
28
29 # A function to do postorder tree traversal
30 def printPostorder(root):
31
32     if root:
33
34         # First recur on left child
35         printPostorder(root.left)
36
37         # the recur on right child
38         printPostorder(root.right)
39
40         # now print the data of node
41         print(root.val),
42
43
44 # A function to do preorder tree traversal
45 def printPreorder(root):
46
47     if root:
48
49         # First print the data of node
50         print(root.val),
51
52         # Then recur on left child
53         printPreorder(root.left)
54
55         # Finally recur on right child
56         printPreorder(root.right)
57
58
59 # Driver code
60 root = Node(1)
61 root.left = Node(2)
```

```
62 root.right = Node(3)
63 root.left.left = Node(4)
64 root.left.right = Node(5)
65 print("Preorder traversal of binary tree is")
66 printPreorder(root)
67
68 print("\nInorder traversal of binary tree is")
69 printInorder(root)
70
71 print("\nPostorder traversal of binary tree is")
72 printPostorder(root)
73
```

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