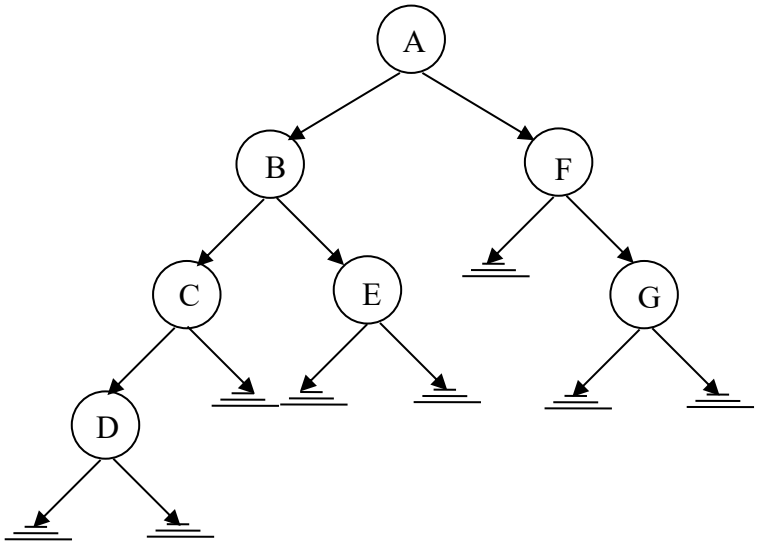
300103 Data Structures and Algorithms

Practical 7
(Week 10)
(2 Marks)

Task 7.1

Base on the following Figure,

a. List the nodes of this binary tree in an inorder sequence.
b. List the nodes of this binary tree in a preorder sequence.
c. List the nodes of this binary tree in a postorder sequence.



Task 7.2

1. Draw a binary search tree with the sequence of the following input:

   55, 79, 90, 25, 110, 40, 85, 52, 30, 45, 65, 48, 98, 50, 80, 58, 70

2. Redraw the binary search tree after inserting a new key 47 into the tree. List all the nodes that are visited.
3. Redraw the binary search tree again after deleting key 79 from the tree.

Task 7.3

Based on the binary search tree ADT implementation: binaryTree.h and binarySearchTree.h, add code to define the function, `leavesCount(binaryTreeNode<elemType> *p)`, to return the number of tree leaves in a given binary tree, in the class binaryTreeType (see file binaryTree.h) and test it with the provided driver MainProgram.cpp.

Note: *Although the code you add is in the base class binaryTreeType, you need the*

extended class bSearchTreeType *to create a binary search tree so that you can insert data into a binary tree.* binaryTreeType *does not contain an* insert *function.*

Task 7.4

Download code Task7_4AVL_Tree.zip. Read the code carefully and answer the following questions to your tutor:

1. The AVLTree structure is defined as a template with two arguments:

```
template <class TYPE, class KTYPE>
class AvlTree
```

   What is the second data type, KTYPE, for? How to use it?

2. What is the functionality of the following two functions?

```
bool AVL_Retrieve(KTYPE key, TYPE& dataOut);
NODE<TYPE>* _retrieve KTYPE key, NODE<TYPE> *root)
```

   How do they work?

3. In the implementation of AVL Traverse, there is a pointer *process.

```
void AVL_Traverse(void (*process)(TYPE dataProc));
void _traversal(void (*process)(TYPE dataProc),
                NODE<TYPE>    *root);
```

   What is this pointer for?