300103 Data Structures and Algorithms

Practical 7

(Week 10)

(2 Marks)

Task 7.1

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 7.2

Download the file *Task7_2BST.zip*. Based on the binary search tree ADT implementation: binaryTree.h and binarySearchTree.h, add code to the class binaryTreeType (see file binaryTree.h) to finish the definition of the function, `leavesCount (binaryTreeNode<elemType> *p)`. The function is to return the number of tree leaves of a binary tree. Test the whole program with the provided driver MainProgram.cpp.

Task 7.3

Download code AVL_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?

Task 7.4
Write a program that reads words from a text file and store occurrence frequency of each word in a STL map. Download the file `AVL_Tree.zip` and redo the task by using the provided AVL tree implementation. Print the AVL tree.

Hint: *Use word as key and the frequency as data.*