1. (60%) In this problem, you will write Java code for simple operations on binary search trees where keys are integers. Assume you already have the following code. Assume the method bodies, even though not shown, are correct and implement the operations as the name suggest.

```
public class BinarySearchTreeNode {

    public int key;

    public SomeType value;

    public BinarySearchTreeNode left;

    public BinarySearchTreeNode right;

}

public class BinarySearchTree {

    private BinarySearchTreeNode root;

    public void insert(int key, SomeType value) { ... }

    public void delete(int key) { ... }

    public SomeType find(int key) { ... } }
```

(A) (20%) Add a method *public int treeHeight()* to the BinarySearchTree class that computes the height of the tree.

(B) (20%) Add method *public void deleteMax()* to the BinarySearchTree class that deletes the maximum element in the tree (or does nothing if the tree has no elements). You may want a helper method.

(C) (20%) For your answers in part (a) and part (b), give a worst-case big-O running time in terms of n, where n is the number of nodes in the tree. Do not assume the tree is balanced, and your answer in (C) needs to match your answer in (A) & (B).

注意:背面有試題

2.　(40%) In Java code, write a linked list class (named LinkedList) where the contained node is of class name Node, and the Node has only one attribute containing an integer, i.e., it is a linked list of integers.　Write the following functions for this class:

(A) (10%) find: Search whether an integer is contained in the list, if yes return the node, otherwise return null

(B) (10%) delete: Given an integer, the node(s) in the list will be deleted if the node's value is equivalent to the given integer

(C) (20%) reverse: Reverse the linked list, which will return another linked list containing the same elements but in exact reverse order