**Node List Tree**

Please build a binary tree’s data structure. The value of each tree node is NodeList. Here is the class of the node list.

1. **public** **class** NodeList {
2. **public** String value;
3. **public** NodeList next;
5. **public** NodeList(String \_value, NodeList \_next){
6. value = \_value;
7. next = \_next;
8. }
10. **public** NodeList(String \_value){
11. value = \_value;
12. next = **null**;
13. }
14. }

And here is the part of the NodeListTree’s class.

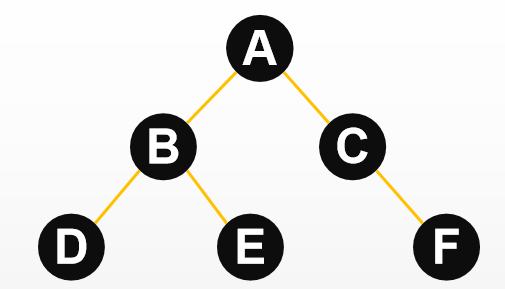
1. **public** **class** NodeListTree {
2. **public** NodeList val;
3. **public** NodeListTree left, right;
4. }

Based on the above codes, please complete the following problems.

**Problem 1:** Please complete the Constructor function of the NodeListTree. You should consider two situations. Situation 1 is the input, including all three data, val, left and right. Situation 2 is that the input only includes the val data. In situation 2, the value of right and left should be null.

**Problem 2:** Please create NodeListTree data, named root in the primary function based on the information below.

The NodeListTree’s overall structure is shown below.



For each node, the node list is shown as follows:

1. >Hello
2. >Everyone
3. >Welcome
4. >To
5. >This
6. >Course

**Problem 3:** Please create a function that can output the pre-order traversal of a given NodeListTree type value. For each NodeList type node, you only need to print the first node’s value. The function should resemble this:

**public static String printFirst(NodeListTree root){}**

Example: The above ‘root’ output should be ‘ABDECF’.

**Problem 4:** Please create a function that can output the medium-order traversal of a given NodeListTree type value. For each NodeList type node, you only need to print the second node’s value. The function should resemble this:

**public static String printSecond(NodeListTree root){}**

Example: The above ‘root’ output should be ‘ToEveryoneThisHelloWelcomeCourse’.

**Problem 5:** Please create a function. Suppose you input a NodeListTree type value and a character type value (a lowercase character) into this function. In that case, it will return an integer number, which indicates the letter appears many times in this NodeListTree type value—disregarding upper and lower case. You can use the code you wrote in Problems 1 to 5. The function should be shown like this:

**public static int returnAns(NodeListTree root, char aim){}**

Example: The output of the (root, ‘c’) should be 3.