Question 1 [40 points]

You are about to write FOUR methods to get information of a person stored in a given Tree in which each Node contains person *ID* and *NAME* as the following example:

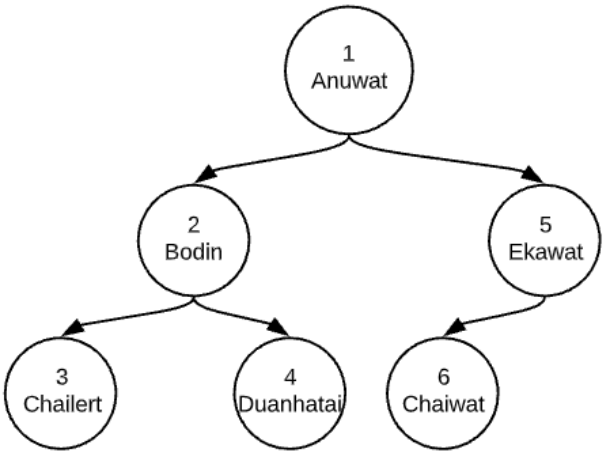

Fig1: Example of the given Tree ts[0].

The package contains **THREE** java files:

- `Node.java`: This is a Node class that can store *ID* and *NAME* as data with links to the left and right child node. **DO NOT MODIFY THIS CLASS!**
- `TreeTester.java`: This is the main program. **DO NOT MODIFY THIS CLASS!**
- `TreeManagement.java:` This Class contains four functions to manage the Tree. **YOU HAVE TO IMPLEMENT THESE FUNCTIONS!**

Please work on the following tasks to complete the program.

Task 1 (10 points): inOrder Navigation

Complete a static method named `printInorderName(Node root, String _result).` This method performs *in-order traversal* and returns all names in the given Tree. The method takes two parameters Node *root* and String *_result,* then returns String of all name in a specific format. For example, by calling the method `printInorderName(ts[0], result),` the result returned from the method will be "Chailert|Bodin|Duanhatai|Anuwat|Chaiwat|Ekawat|".

Running `printInorderName` method in `TreeTester.java` should give the following output:

```
@TASK1
List of All person in Tree=>Chailert|Bodin|Duanhatai|Anuwat|Chaiwat|Ekawat|
List of All person in Tree=>Wyatt|Jackson|Mason|Carter|Amily|Grayson|Harper|Adison|Avery|Jaaxon|
```

Task 2 (10 points): Count the number of all name begin with the given character

Complete a static method `countName(Node root, char _char).` This method performs navigation and counting the number of persons whose name begins with the given parameter *_char*. The method takes Node *root*, and char *_char* as parameters, then returns the integer number of all names that begin with the character _char. For example, by calling the `countName(root, 'C')` method, the result will be an integer of 2 (by counting person name: Chailert and Chaiwat).

Running `countName` in `TreeTester.java` should give the following output:

```
@TASK2
Number of person whose name begins with 'C' = 2
Number of person whose name begins with 'A' = 3
```

Task 3 (10 points): Find ID by the given Name (MUST BE WRITTEN AS RECURSIVE FUNCTION)

Complete a static method `searchIDByName (Node root, String _name)` to find the ID of a specific person from the given name. The method takes a Node *root*, and String *_name* as parameters, then return the integer ID of the matched name. For example, by calling the method `searchIDByName(Node root, "Duanhatai")`, the result will be an integer of ID= 4.

Running `searchIDByName` in `TreeTester.java` should give the following output:

```
@TASK3
The person named Duanhatai has id = 4
The person named Avery has id = 112
```

Task 4 (10 points): Find the longest name from the given Tree (MUST BE WRITTEN AS RECURSIVE FUNCTION)

Complete a static method `findLongestName(Node root)` to find a person who has the longest name in the Tree. The method takes a Node *root* as a parameter and then returns the Node of a person who has the longest name in the Tree. For example, by calling the method `findLongestName(Node root)`, the method must return the Node of person name "Duanhatai".

Running `findLongestName` in `TreeTester.java` should give the following output:

```
@TASK4
Duanhatai has the longest name in the Tree
Jackson has the longest name in the Tree
```

A File to submit: TreeManagement.java

☺ ~ ~ GOOD LUCK ~ ~ ☺