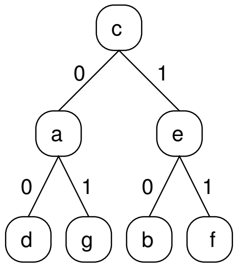
Lab 10: Tree

```
Location: CourseWeb -> Labs/Recitations -> Lab 10: Tree
Download the following files:
  1. ShowPath.java
  2. BinaryNodeInterface.java
  3. BinaryNode.java
  4. SimpleQueue.java
```

Introduction

In this lab, you are going to practice traversing a tree using recursion with a little of back-tracking. Recall that a **path** begins at the root and goes from node to node along the edges that join them. Suppose we name a left edge `'0'` and the right edge `'1'` as shown in Figure below:



In doing so, we can represent a path by a string of 0s and 1s. For example, the path to the node containing `'g'` is `"01"`, the path to the node containing `'b'` is `"10"`, the path to the node containing `'c'` is `""` (empty string), and so on.

What to Do?

For this lab, you are going to implement two methods marked `TO DO` in `ShowPath.java`. In `ShowPath.java`, a tree of random unique characters is constructed using the method `generateTree()`. This tree consists of characters from `'a'` to `'z'` and from `'A'` to `'Z'`. The method `generateTree()` will return the root node (of type `BinaryNodeInterface<Character>`) of the tree of characters. Note that a series of `BinaryNode` can be viewed as a tree. So, for this lab, you are going to work with a series of `BinaryNodes` instead of a tree. The two methods that you have to implement are as follows:

1. `String getAllPaths(final BinaryNodeInterface<Character> root)`: This method simply returns a string representing all paths to every nodes of the tree root at `root`. The returned string **must** be in the following format:

```
r
d 0
U 00
t 000
```

```
I 0000
Y 00000
P 00001
M 0001
T 00010
:
:
b 110
h 1100
Q 1101
y 111
O 1110
s 1111
```

Note that the format for each line must be a character, follows by a single space, follows by the string representing the path from the root note to the node containing the character, and follows by the newline character (`'\n'`). If your format is incorrect, the tester may report incorrect test result. The above example is done according to **Preorder Traversal**. However, you can use any type of traversal. Note that **YOU MUST** use a type of traversal. You are **NOT ALLOWED** to use the method `getPathTo()`.

2. `String getPathTo(final BinaryNodeInterface<Character> root, char c)`: This method returns a string representing a path from the root to the node containing the character `c`.

Test Class

Methods to test your implementations are included in `ShowPath.java`. This will test your implementation of methods `getAllPaths()` and `getPathTo()` and show your total points (out of 10). If you do not get the full 10 points, you should keep trying to fix your code until you get 10 points. **Note** that these test methods are not perfect. It cannot tell you why your program is incorrect. You may have to look at the source code of `checkGetAllPaths()` and `checkGetPathTo()` and see why it says `FAIL` and trace your code.

Due Date and Submission

For the due date, please check the lab in the CourseWeb. Submit your `ShowPath.java` to the CourseWeb under this lab by the due date. **No late submission will be accepted**.