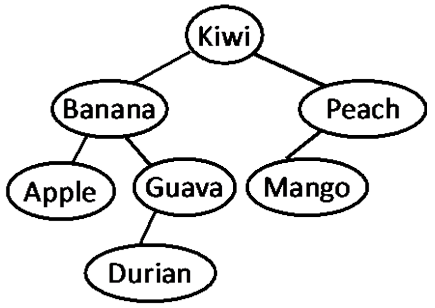
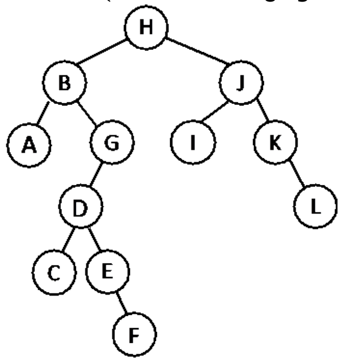
CSSE230 Exam 2 – Winter 2018: Sections: 03 & 04

Overview:

- Work in the file `BinarySearchTree.java` found in the **Exam2** Eclipse project downloaded from Moodle
- When finished, upload `BinarySearchTree.java` to the CSSE230 Moodle site, the *Exam 2 – Computer Part* assignment
- *Reference*: The two trees below are used in some of the JUnit test cases (see `Testing.java`)



Fruit Tree Example



ABC Tree Example

- To simplify this exam, the BST used in `BinarySearchTree.java` is a BST of `String`
- You are required to use recursion in your implementation. There are three methods for class **BinarySearchTree**. For each of these you are permitted to write recursive helper operations which can reside in either the **BinarySearchTree** class or in the **BinaryNode** class.
- You may add methods to the BinaryNode class, but not add/remove/change any of its fields

Performance Requirements:

- If the operation to be implemented is a BST "traversal" operation, then it must run in O(N) time and traverse the tree no more than one time, where N = number of nodes in the tree
- If it is a "navigation" operation, it must run in O(Height(T)) time and navigate to its destination no more than once

The Required Operations:

1. (16 points) – Implement `int countNodesWithOddLengthLabels()` whose job is to count and return the number of nodes in the tree that have a label (i.e., the String stored as data in the node) whose length is an odd number.
   - For the *Fruit Tree* example, the answer should be 4 (by counting "Apple", "Guava", "Mango", and "Peach")
   - For the *ABC Tree*, it should be 12.

2. (20 points) – Implement **`void`** `deleteNodesWithOneChild()` whose job is to remove all nodes from the BST that have only one child. The resulting tree must be a BST containing all the nodes from the original tree that had either zero or two children.
   - For the *Fruit Tree* example, the nodes labeled with "Guava", and "Peach" should be removed.

3. (20 points) – Implement `ArrayList<String> getDirectionsToItem(String x)` whose job is to return an ArrayList of String containing navigation directions to the value passed in as a parameter to 'x'.
   - For the *ABC Tree*, and "D" as the parameter, the result should be:
     `[left at H, right at B, left at G, D found]`
   - For the For the *Fruit Tree*, and "Asian Pear" as the parameter, the result should be:
     `[left at Kiwi, left at Banana, right at Apple, Asian Pear not found]`

Testing:

- Utilize the JUnit tests provided in the file `Testing.java`