HackerRank
For Work

Spring_2018_CSYE7200_Fi... 60 minutes

Question - 1
Zip the Stream

SCORE: **18 points**

Implement all TODOs.
Please note:

1. Actual compiling/running code is required for this question, grading will base on test cases. Uncompiled code will receive no more than 50% of the score.
2. If you can't figure out how to implement one of the TODO, just leave it as is, do not change it, make sure your code compiles.
3. Don't worry if you can't get one of the todo right, the test case are designed as even you only implement one todo, you still get points for that part.

Question - 2
DFS

SCORE: **28 points**

Implement a depth-first-search method using tail recursion. You are given a trait *BinaryTree* with two case classes: *Node* and *Empty*. A *Node* has a value and two child *Nodes*, marked *left* and *right*. In a binary tree, a node's left sub-tree contains values which all compare *less-than* the node's value, while the right sub-tree contains values which all compare *greater-than-or-equal* to the node's value. Depth-first search, as opposed to breadth-first-search, is a method of tree (or acyclic graph) traversal in which we recurse through the sub-trees before dealing with the node itself. Note that, in the case of a binary tree, we could traverse it in value-order by recursing into the left sub-tree, then dealing with the value, then recursing into the right sub-tree. But that's not what is being asked for here, and so not how the unit tests are specified.

The depthFirst method takes a function *f: T=>U* and will apply that function to every value in the order determined by the DFS traversal.

Recall that when we create a tail-recursive inner method, we usually have two parameters: one represents the work yet to be done, the other represents the work that has been done.

Note: Although actual compile and running is strongly recommended, this coding question will also be graded manually.