CS 300: ADVANCED PROGRAMMING

SPRING 2024

Module: Haskell

Programming Assignment 1.2

Syed Babar Ali
School of Science and Engineering

Deadline: Feb 17, 2024 (Saturday) 7 PM

Assignment Guidelines

- The assignment is due at **7 PM on Feb 17, 2024 (Saturday)**
- There are a total of **7 questions.**
  - You have to attempt **all of the 3 questions from Medium category**.
  - You have to attempt **3 out of 4 questions from Hard category.**
- Total marks of the assignment are **39**.
- **No imports allowed in any of the questions.**
- You are allowed to use take, drop, head, tail, div, rem, + , - , * , !!, ++, where, otherwise, let, guards, : , → , ← , if-then-else, &&, ||, not, ==, and list compression.
- You can add Eq and/or Ord to the function signatures if you want to.
- We have provided you with a haskell file **"code.hs"** with **function signatures** and **some test cases.**
- Please note that the test cases may not cover all the cases for the questions. There may be other (corner) cases which students have to figure out themselves.
- There are **NO** grace days for any of the assignments in this course.
- There is **NO** late days policy. No submission will be accepted after the deadline.
- **Plagiarism is strictly prohibited.** Students are not allowed to discuss their solutions with others or look over the internet. However, they may seek help from the course staff **ONLY**.
- Students may discuss their queries or clear their confusions about the questions in office hours or on slack (make use of the assignment channel).
- There will be **NO partial** marking for any of the questions. This is being done to be fair to all the students in this course. However, the questions will be graded according to the number of test cases that are passing. For example: if a 10 marks question has 10 test cases and 5 of them are passing for a student, then they will get 5/10 for that question.
- The **submissions will be done using GitHub Classroom**. Your **latest commit** in your assignment repository before the deadline will be considered for the final submission.
- You will have to install hspec which is a testing framework for haskell that we are using for this assignment. It can be simply installed using the following command:
  - "cabal update && cabal install --lib hspec"
- The evaluations may consist of vivas, shortly after the deadline.
- **Start Early & Happy Coding!**

Questions Category: Medium (5 Marks Each)

Question 1: Letter Combination of a Phone Number

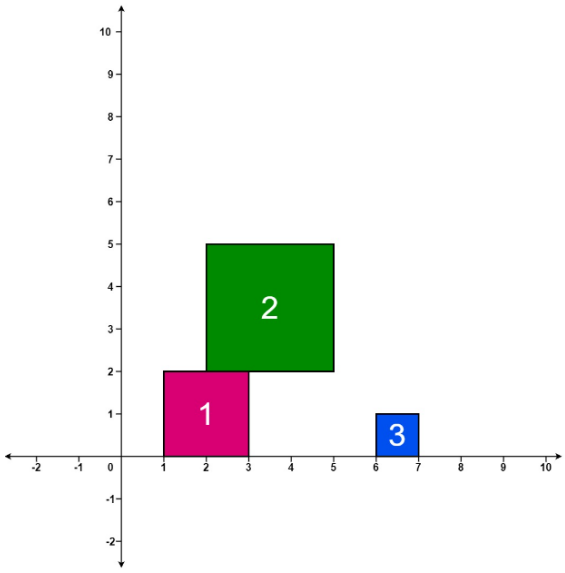
Write a function that takes a string as input, which contains only digits ranging from 2 to 9. The function should return a list of all possible letter combinations that these digits could represent on a telephone keypad. The keypad follows the classic mapping where each digit is associated with certain letters (e.g., 2 corresponds to 'a', 'b', 'c'; 3 to 'd', 'e', 'f', etc.). The output should be arranged in such a way that it starts with all the combinations that can be formed using the first digit, followed by all the combinations using the second digit, and so on, for each digit in the input string.

Sample Input: digits = "2"
Output: ["a", "b" , "c"]
Sample Input: digits = "24"
Output: ["ag", "ah" , "ai", "bg", "bh" , "bi", "cg", "ch" , "ci"]

Question 2: Height of a Falling Square

There are several squares being dropped onto the X-axis of a 2D plane.
You are given a 2D integer array positions where positions[i] = [`left_i`, `sideLength_i`] represents the ith square with a side length of `sideLength_i` that is dropped with its bottom left edge aligned with X-coordinate `left_i`. Each square is dropped one at a time from a height above any landed squares. It then falls downward (negative Y direction) until it either lands on the top side of another square or on the X-axis. A square brushing the left/right side of another square does not count as landing on it. Once it lands, it freezes in place and cannot be moved.
For each square dropped, you must record its height.

Return an integer array ans where ans[i] represents the height described above after dropping the ith square.



Sample Input: positions = [[1,2],[2,3],[6,1]]
Output: [2,5,1]

Explanation:
After the first drop, square 1 lands on the X-axis and its height is 2.
After the second drop, the tallest stack is squares 1 and 2 with a height of 5.
After the third drop, square 3 lands on the X-axis and its height is 1.
Thus, we return [2, 5, 1].

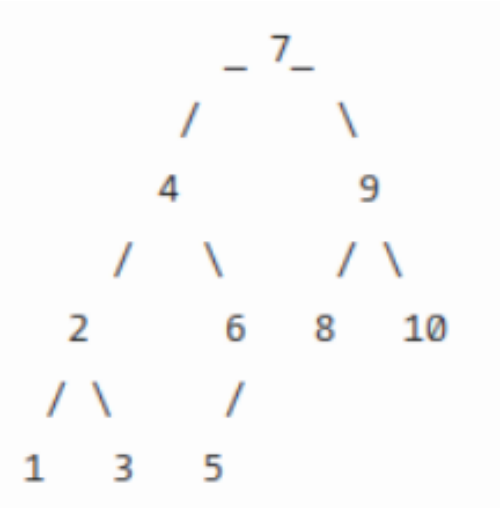
Sample Input: Positions = [[100,100],[200,100]]
Output: [100,100]
Explanation:
After the first drop, square 1 lands on the X-axis, at x=100, and its height is 100.
After the second drop, square 2 lands on the X-axis, at x=200, and its height is 100. Note that square 2 only brushes the right side of square 1, which does not count as landing on it.
Thus, we return [100, 100].

Question 3: Post-Order to Breadth-First Conversion

According to Wikipedia, a complete binary tree is a binary tree "where every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible."

The Wikipedia page referenced above also mentions that "Binary trees can also be stored in breadth-first order as an implicit data structure in lists, and if the tree is a complete binary tree, this method wastes no space."

Your task is to write a function that takes a list of integers and, assuming that the list is ordered according to a **post-order traversal** of a complete binary tree, returns a list that contains the values of the tree in breadth-first order. Example 1: Let the input list be [1, 3, 2, 5, 6, 4, 8, 10, 9, 7]. This list contains the values of the following complete binary tree



Output: The output of the function shall be a list containing the values of the nodes of the binary tree read top-to-bottom, left-to-right. In this example, the returned list should be: [7, 4, 9, 2, 6, 8, 10, 1, 3, 5]

Questions Category: Hard (8 Marks Each)
Attempt any 3 of the 4 questions

Question 1: Valid Number
A valid number can be split up into these components (in order): a decimal number or an integer
A decimal number can be split up into these components (in order):

1. A sign character (either '+' or '-').
2. One of the following formats:
   - One or more digits, followed by a dot '.'.
   - One or more digits, followed by a dot '.', followed by one or more digits.
   - A dot '.', followed by one or more digits.

An integer can be split up into these components (in order).

1. A sign character (either '+' or '-').
2. One or more digits.

For example, all the following are valid numbers: ["2", "0089", "-0.1", "+3.14", "4.", "-.9"], while the following are not valid numbers: ["abc ","1a ","1e ","--6", "-+3", "95a54e53", "."].
Given a string, determine whether the string contains a valid number. Note that the string may consists of spaces.

Question 2: Swapping Nodes of a Tree
In this question, we want to swap any two nodes in a tree. Two nodes A and B are swapped if the value of A is an integral multiple of the value of B. You can assume that if A is an integral multiple of B, there is no other node C in the tree for which A is an integral multiple of C, or C is an integral multiple of A. Swaps can be performed between A and B based on the following rules:

- If there is an ancestor-descendant relation between A and B (A is an ancestor of B or B is an ancestor of A), then swap their values only.
- Otherwise, swap the values of A and B as well as the sub-trees (descendants of A and descendants of B).

In the event of multiple swaps, follow these guidelines: Prioritize the swap involving the node at the lowest level. For swaps at the same level, execute them in a left-to-right sequence, starting with the swap that includes the leftmost node at that level. Remember that root is at the lowest level.
The function should perform these swaps throughout the tree and return the

modified tree. The swapping should meet this condition for every pair of nodes that are swapped.

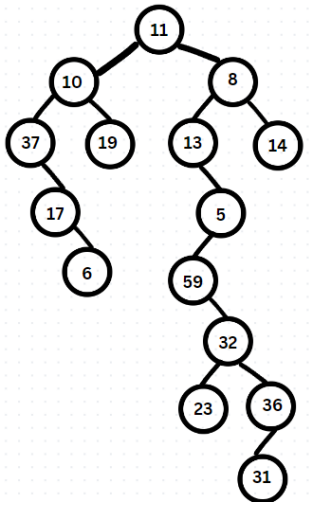


Figure 1: Original Tree

As you can see, there are 3 possible swaps (10 and 5, 8 and 32, 6 and 36). First, we have to do the swap that involves node at the lowest level. In our case, there are two swaps (10 and 5, 8 and 32) that involve nodes (10 and 8) at the lowest level. So, according to our rule, first, we will perform the swap that includes the leftmost node, which in this case is 10.

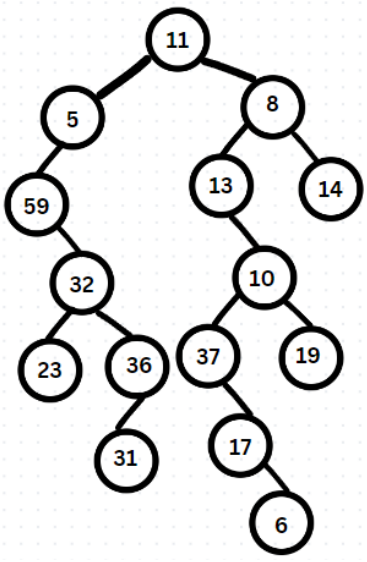


Figure 2: Tree after swapping 10 and 5

Now, we have to do two more swaps (8 and 32, 6 and 36). According to our rule, we perform the swap that involves node at the lowest level, which in this case in 8.

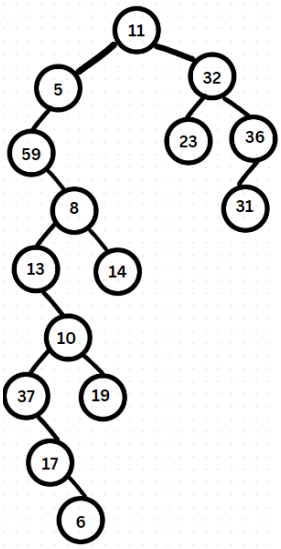


Figure 3: Tree after swapping 8 and 32

Finally, we swap 6 and 36.

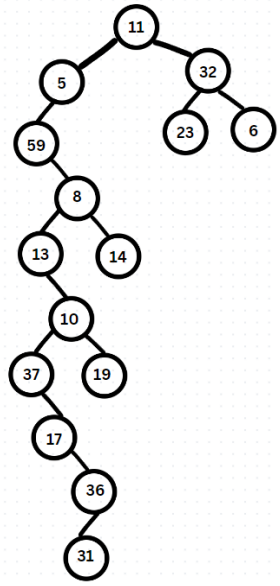


Figure 4: Tree after swapping 6 and 36

Here is another example.

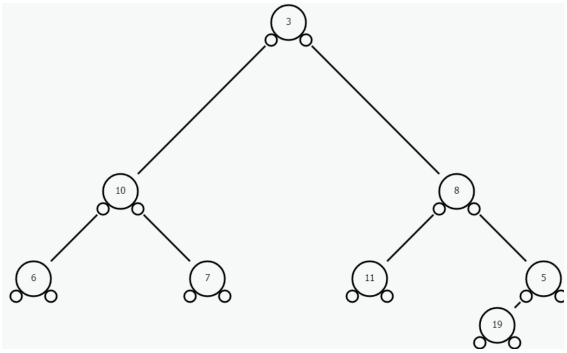


Figure 5: Original Tree

There are 2 possible swaps (3 and 6, 10 and 5). First, we swap 3 and 6 because 3 is at the lowest level. Because there's a parent-child relationship between 3 and 6, we will only swap their values and not the sub-trees.

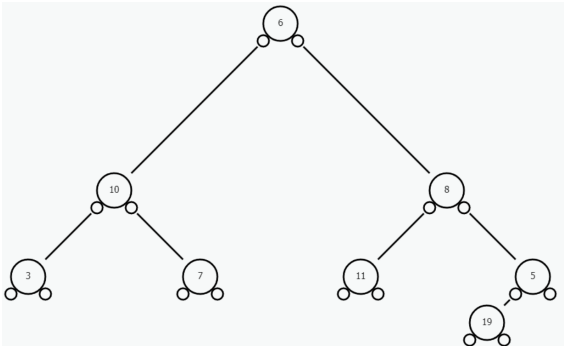


Figure 6: Tree after swapping 3 and 6

Next, we swap 10 and 5.

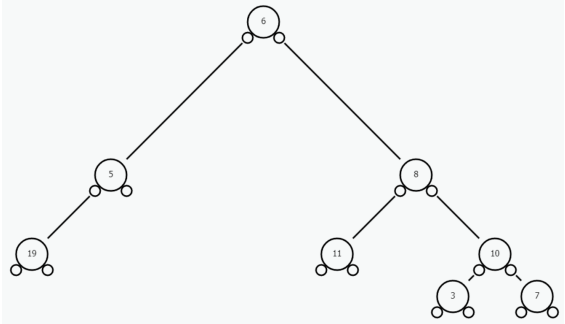


Figure 7: Tree after swapping 10 and 5

Question 3: Basic Calculator

Given a string which represents a mathematical expression, evaluate this expression and return its value. Integer division should truncate towards zero. The possible operators include addition, subtraction, multiplication, and division.

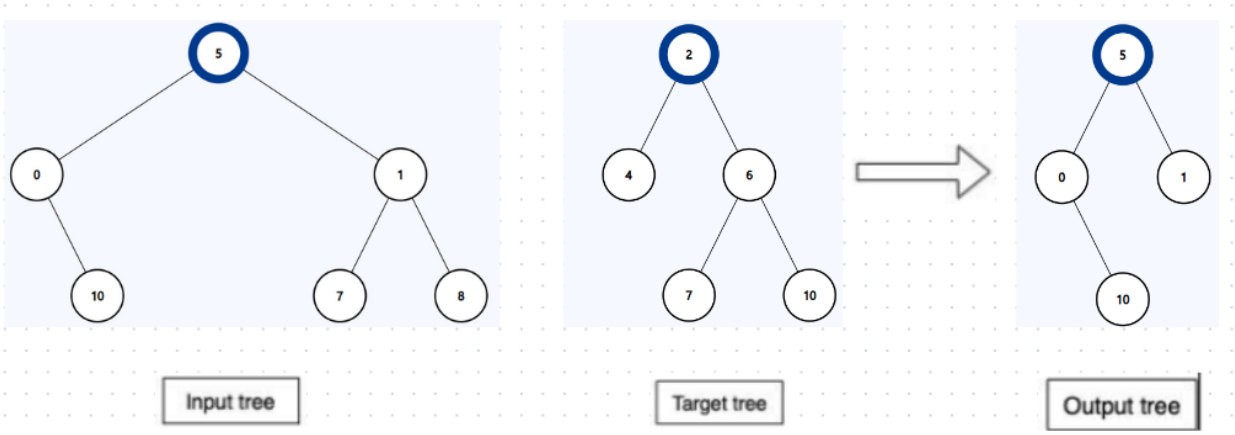
Sample Input: string = "2 + 9/ 3"
Output: 5
Sample Input: string = "(3 + 12) / 5"
Output: 3
Sample Input: string = " 2 * 4 / 3"
Output: 2

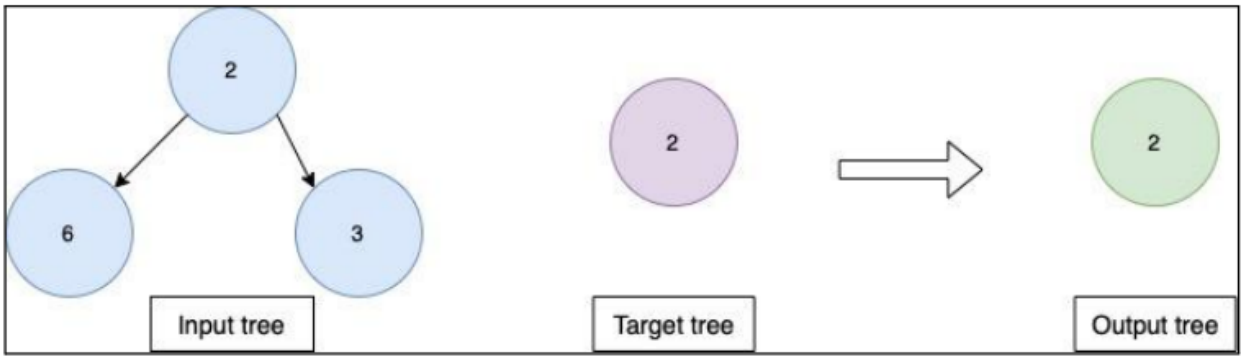
Question 4: Path Sum Synchronizer

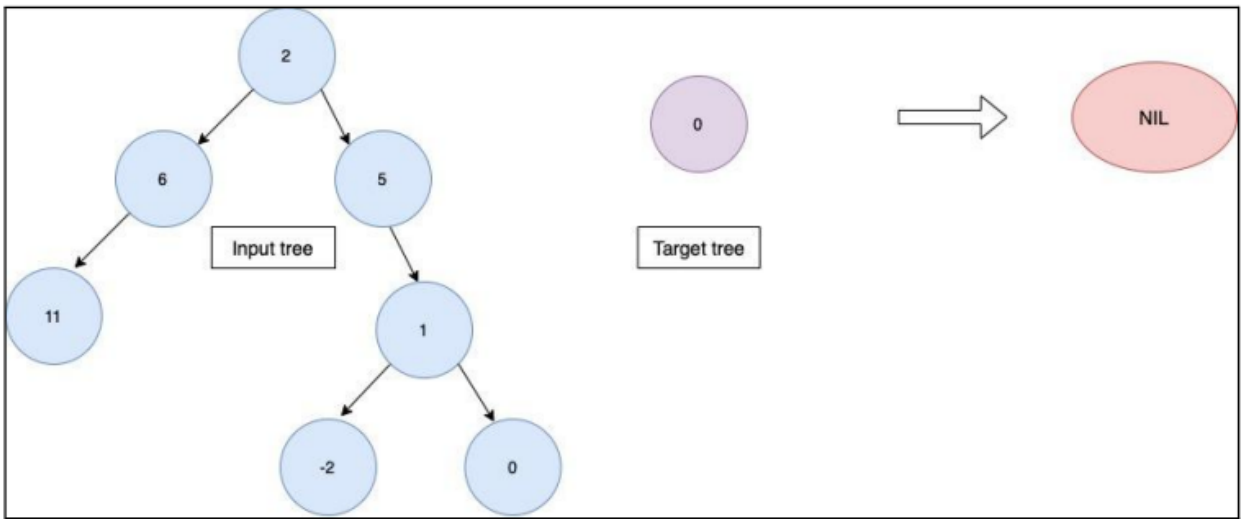
The most general (also the easiest) form of path sum is where, given a binary tree and an integer n, you need to find if there exists a path from the root node to the leaf node which has a sum equal to n. Some hard versions include calculating all the paths that exist from the root node to any node provided the sum equals the target integer (n). However, we will be doing a harder version of this question.

You are provided with two binary trees: the 'target tree' and the 'path tree'. For the target tree, consider the sum of values from the root node to each leaf node as a set of target sums. Your task is to modify the path tree such that it retains only those paths whose sum of values matches any of the target sums derived from the target tree. Essentially, you need to truncate the path tree to include only paths with sums equal to the sums of paths from the root to the leaves in the target tree. Note that the root of the original path tree must be the root of the output tree, if the output tree isn't Nil.

Following are some examples:


5
0
1
10
7
8
Input tree
2
4
6
7
10
Target tree
5
0
1
10
Output tree


2
6
3
Input tree
2
Target tree
2
Output tree


2
6
5
Input tree
11
1
-2
0
0
Target tree
NIL