Advanced Programming Spring 2025

Assignment 1

LUMS
A Not-for-Profit University

Release Date: Feb 7, 2025
Due Date: 11:59 PM, Feb 21, 2025

Assignment Guidelines:

- The assignment is due at **11:59 PM** on **21st February, 2025**. Please note that the deadline will not be extended since the exam for Haskell module will be on the **very next day**.
- There are a total of **15 questions**. You only have to attempt **any 3** out of the **4** hard questions. Total marks of the assignment are **64**.
- Along with the manual, we have provided you with a haskell file, ‘code.hs’, with type signatures and, for your convenience, some test cases. Do not change the name of the file.
- Please note that the test cases **MAY NOT** cover all the cases for the question. There may be other (corner) cases which students have to figure out themselves. Grading will be done based on unseen (hidden) test cases.
- No imports are allowed for any of the questions unless stated otherwise. **Do not** change any of the function signatures or constructors given.
- There will be **NO grace days or late days** for this assignment.
- **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 assignments channel).
- 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 made on Github. Your latest commit in your assignment repo will be considered your final submission. **You will not be able to make changes after the deadline has passed**.
- The evaluations may consist of **vivas**.
- 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 hspec”

- To run the test cases, simply load your haskell file and call the main function (enter “main” in terminal).
- Start early, and good luck!!! 🤡

Assignment Questions Category: Easy 🎂 🎂 🎂(3 marks each)

Question 1 - Matrix Multiplication

You have to perform matrix multiplication of two 2D matrices of arbitrary dimensions and return the result. If the dimensions are incompatible, return [].

Example:
Input:

- Matrix 1: [[1,2], [2,-2]]
- Matrix 2: [[1,2,0], [2,3,0]]

Output: [[5,8,0], [-2,-2,0]]

Type signature:

```
matrixMultiplication :: [[Int]] -> [[Int]] -> [[Int]]
```

Question 2 - Stats of a LinkedList

Given a linked list, return its mean and median in the form of a linked list (first element is mean, second is median).
For median: If the list is odd, return the middle value. If the list is even, return the rounded down average of the two middle values. Return 0 if the list is empty.
For mean: Return the rounded down average of all the elements in the list

Example:
Input: ListNode 3 (ListNode 5 (ListNode 1 (ListNode 9 Null)))

Output: ListNode 4 (ListNode 4 Null)

Type signature:

```
listStats :: LinkedList Int -> LinkedList Int
```

Question 3 - Largest Adjacent Sum

Find the three adjacent numbers with the largest sum in a list. Return the three numbers in the order in which they are in the list and in the form of a tuple. If there are multiple tuples with the largest sum, return the tuple which appears first in the list. Assume that there are at least three elements in the list.

Example:
Input: [2,4,1,6,4,3,7,2]

Output: (4,3,7)

Type signature:

```
largestAdjacentSum :: [Int] -> (Int,Int,Int)
```

Question 4 - Collatz Conjecture

The Collatz Conjecture is a famous mathematical sequence defined as follows:

● If n is even, the next number is n/2.

● If n is odd, the next number is 3n+1.

● This process repeats until the sequence reaches 1.

Your task is to implement a function that takes a positive integer n and returns a tuple containing:

1. The number of steps required to reach 1.
2. The maximum value encountered in the sequence before reaching 1.

Example:
Input: 6

Output: (8,16)

Explanation: The sequence is: 6->3->10->5->16->8->4->2->1
- **Steps:** 8 steps to reach 1
- **Max Value:** 16 is the highest number encountered

Type signature:

```
collatzConjecture :: Int -> (Int,Int)
```

Question 5 - Array Product Except Self

Write a Haskell function that takes a list of integers and returns a list answer where answer[i] is equal to the product of all the elements of nums except the one at index i.

Constraints:
The input list may contain negative numbers and zeros.
The length of the list is at least 2.

Example:
Input: [2,3,4,5]

Output: [60,40,30,24]

Type signature:

```
productExceptSelf :: [Int] -> [Int]
```

Question 6 - Longest Consecutive Sequence

Write a function that takes a list of integers as input and returns the length of the longest consecutive sequence of **prime** numbers.

Example:
Input: [4,7,2,3,5,10,11,13,17,6]

Output: 4

Type signature:

```
longestPrimeSeq :: [Int] -> Int
```

Assignment Questions Category: Medium 🤔(5 marks each)

Question 7 - Leaf Deletion

You will be given a Binary Tree containing strings. Your task is to delete all **leaf** nodes that are **NOT** a substring of their parent. If there is only a single node in the tree, return Nil.

Example:
Input: TreeNode (TreeNode (TreeNode Nil "act" Nil) "cat" (TreeNode Nil "ca" Nil)) "root" (TreeNode Nil "to" Nil)

Output: TreeNode (TreeNode Nil "cat" (TreeNode Nil "ca" Nil)) "root" Nil

Explanation: *"act"* is a leaf node and it is not a substring of its parent, *"cat"*, therefore it is deleted. Similarly, *"to"* is a leaf node and it is not a substring of its parent, *"root"*, therefore it is deleted. *"ca"* is a leaf node and it is a substring of its parent, *"cat"*, therefore it is not deleted.

Type signature:

```
leafDeletion :: Tree String -> Tree String
```

Question 8 - Text Editor

You are building a simplified text editor that processes keystrokes in real-time. Given an input string representing a sequence of typed characters, simulate how the text would appear after all keystrokes have been processed.

The special character '#' represents a backspace, meaning it deletes the most recently typed character (if any). Consecutive backspaces should continue deleting characters, and any leading backspaces should be ignored. The character '@' represents an undo operation, which restores the most recently deleted character (if any). Undo operations do not affect normal typed characters—only characters removed by '#' can be restored. If there are no deleted characters left to restore, '@' does nothing.

Example:
Input: "text#edi@tor"

Output: "texedittor"

Explanation:

1. **"t", "e", "x", "t"** → typed as normal.
2. **"#"** → deletes the most recent character **"t"** → **"tex"** remains.
3. **"e", "d", "i"** → typed as normal → **"texedi"**.
4. **"@"** → restores the most recently deleted character (**"t"**) → **"texedit"**.
5. **"t", "o", "r"** → typed as normal → **"texedittor"**.

Type signature:

```
textEditor :: String -> String
```

Question 9 - Halki si OS

You are given a string that represents an absolute path in a Unix-like file system. This path will always begin with a forward slash /. Your task is to process this path and produce a normalized (canonical) version.
Behavior of a Unix File System
1. A single period (".") represents the current folder and should be ignored during normalization.
2. A double period ("..") indicates moving up one directory level to the parent folder.
3. Multiple consecutive slashes ("//", "///", etc.) should be treated as a single slash.
4. Any sequence of periods not matching the above (e.g., "...", "....") is treated as part of the directory or file name, not a special command.
Normalization Rules
● The normalized path must always begin with a single slash ("/").
● All directories and files within the path should be separated by exactly one slash.
● The resulting path should not end with a slash unless it is the root directory ("/").
● Any invalid folder references (e.g., "." or ".." beyond the root level) should be resolved or ignored during normalization.
Your objective is to implement a function that returns the normalized path.

Example 1:
Input: "/projects/reports//../images"

Output: "/projects/images"

Explanation: *".."* moves one level up, skipping the "*reports*" directory. Double slash is treated as a single slash.

Example 2:
Input: "/archive/…/temp/../data/."

Output: "/archive/…/data"

Explanation: *"..."* is treated as a valid folder name, while *".."* and *"."* are resolved as per the above rules.

Type signature:

```
halkiOs :: String -> String
```

Question 10 - Palindromic Paths with Minimum Swaps

Given a list of strings, determine the minimum number of adjacent swaps needed to rearrange each string into a palindrome. If a string cannot be rearranged into a palindrome, return -1 for that string. The final result should be the sum of the minimum swaps required for all strings in the list.

Example 1:
Input: ["arcerac", "banana", "bbo"]

Output: 2

Explanation:
"**arcerac**": swapping the *"a"* and *"c"* on the right (1 swap), then swapping *"c"* and *"r"* (another swap), results in a total of **2** swaps.
"**banana**": cannot form a palindrome, so it contributes **-1**.
"**bbo**": Swapping *"o"* and *"b"* forms a palindrome with **1** swap.
The sum of the tree swaps would then become 2 + (-1) + 1 = 2.

Type signature:

```
palindromeSwaps :: String -> Int
```

Question 11 - Maximum Streak of Consecutive Numbers

Write a function that takes a list of integers and returns the length of the longest sequence of consecutive numbers (numbers appearing in increasing order without gaps).
The numbers in the sequence do not need to be adjacent in the original list. The list can contain positive and negative numbers. The list may not be sorted.

Example 1:
Input: [100,4,200,1,3,2]

Output: 4

Explanation: 1,2,3 and 4 form the longest sequence of consecutive numbers.

Type signature:

```
maxStreak :: [Int] -> Int
```

Assignment Questions Category: Hard 🤡 (7 marks each)

Question 12 - Tree Deduction

Given a binary tree, transform each node's value to the difference between the sum of its original right subtree and the sum of its original left subtree. After transforming all nodes, recursively remove all leaf nodes with a value of zero until no such leaves remain.

- The sums for each node are computed based on the original tree's structure, not the transformed or pruned tree.
- A leaf is a node with no left and right children.
- If the tree becomes empty after pruning, return an empty tree.

Pruning Process:

- Remove all zero-valued leaves (nodes with no children) after the entire tree is transformed.
- This pruning is repeated until no zero-valued leaves remain. For example, pruning a leaf may turn its parent into a new leaf, which must then be checked for the condition.

Example 1:
Input: (TreeNode (TreeNode (TreeNode Nil 4 Nil) 2 (TreeNode Nil 5 Nil)) 1 (TreeNode (TreeNode Nil 6 Nil) 3 (TreeNode Nil 7 Nil)))

Output: (TreeNode (TreeNode Nil 1 Nil) 5 (TreeNode Nil 1 Nil))

Explanation:
For element **1**, the right subtree sum = 16 and the left subtree sum = 11. So the node is altered to 16-11=5.
For element **2**, 4 - 5 = 11
For element **3**, 7-6 = 1

Example 2:
Input: (TreeNode (TreeNode (TreeNode Nil 5 Nil) 2 (TreeNode Nil 5 Nil)) 1 (TreeNode Nil 3 Nil))

Output: TreeNode Nil (-9) Nil

Explanation:
For element **1**, the right subtree sum = 3 and the left subtree sum = 12. So the node is altered to 3-12=-9.
For element **2**, 5 - 5 = 0
For element **3**, 0-0 = 0

Now, **2** and **3** have both been reduced to **0**. Therefore, we have to prune the tree, i.e. remove zero-valued leaves.

Type signature:

```
treeDeduction :: (Eq a,Num a) => Tree a -> Tree a
```

Question 13 - Citywide Poison Spill

You are tasked with assessing the impact of a toxic spill that started from various locations in a city. The city has several poisonous containers, each with specific coordinates and a spread radius that determines how far the poison can travel. When one container's poison spills, it can trigger other containers within its spread radius, causing a chain reaction. The task is to find the maximum number of containers that will be affected by the poison if you activate any one of them. You can map out each container's poison damage area and deduce how many subsequent containers can be affected from a specific container.

You are given a list of **poisonous containers**, where each container is represented as a **tuple**:

(x,y,r)

- x,y represent the **coordinates** of the container.
- r represents the **spread radius**—how far the poison can travel in any direction (up, down, left, right, or diagonal) to trigger other containers.

Two containers are triggered if the Euclidean distance between them is less than or equal to the spread radius of the activating container.

Example 1:
Input: [(0, 0, 3), (2, 0, 2), (4, 0, 2), (6, 0, 2), (8, 0, 1)]

Output: 5

Explanation: Container **0** (at (0,0) with radius 3) can trigger **1** (2,0) and **2** (4,0). **1** (2,0) can trigger **2** (4,0). **2** (4,0) can trigger **3** (6,0). **3** (6,0) can trigger **4** (8,0).

Type signature:

```
poisonSpill :: [(Int,Int,Int)] -> Int
```

Question 14 - Bilal Trapped in a Maze

Bilal finds himself trapped in a complicated maze. The walls are high, and the pathways are narrow. He can only move up, down, left, or right, while avoiding the solid stone barriers that block his way. The maze is represented as a 2D list where each cell is either open ('O') or blocked ('X'). Two special cells are present: the start cell, where Bilal currently is, marked as 'S' and the goal cell marked as 'G'.

Create a function that takes this maze as input and returns a list of directions representing a path for Bilal from the start to the goal. The path should navigate through open cells and avoid blocked cells. Find the shortest path (you can assume there is only one); if there is none return an empty list.

Example 1:

```
Input: [ "SOOOO"
       , "OXXXO"
       , "OOOXO"
       , "XXOXO"
       , "OGOOO"
     ]
```

Output: [South, South, East, East, South, South, West]

Explanation: You can go [South, South, East, East, South, South, West], or you can go [East, East, East, East, South, South, South, South, West, West, West]. However, the second path is longer, therefore you choose the former path.

Example 2:

```
Input:
 [ "SOOO",
   "OOXO",
   "OXOX",
   "OXOX",
   "OOOG"
 ]
```

Output: [South,South,South,South,East,East,East]

Type signature:

```
mazePathFinder :: [[Char]] -> [Direction]
```

Question 15 - Halloween Escape

You are the last survivor in a deadly chase by Michael Myers in Haddonfield. You are currently hiding in a specific room of a specific house, and you need to escape. You are given a list of available hiding spots, each represented by a tuple, where the first element is the house number and the second element is the room in that house. The same hiding spot can appear in the list more than once.

You have to find the **minimum number of moves** needed to go from your current hiding spot (the first element in the list) to the last hiding spot in the list. The movement rules are as follows:

1. You can move to the next hiding spot in the list.
2. You can move to the previous hiding spot in the list.
3. If you are in a specific room in a house, you can **teleport** to the same room in a different house.

Example:

Input: [("House 1", "Closet"), ("House 1", "Basement"), ("House 2", "Closet"), ("House 3", "Attic"), ("House 3", "Bathroom"), ("House 2", "Basement"), ("House 2", "Attic")]

Output: 3

Explanation:
- You start at House 1's closet. You can teleport to House 2's closet.
- You can then move to House 3's attic, which is the next hiding spot in the list.
- You can now teleport from House 3's attic to House 2's attic. You have reached the end.

Every other way to reach from the start to the end would require either more or an equal number of moves.

Type signature:

```
halloweenEscape :: [(String, String)] -> Int
```

End of Assignment 1