Reflection on Data Structures Adventure

Student Details

- Name: Basil Moyikkal
- Student ID: C0908488

---

1. The Array Artifact

What I Learned

In this challenge, I learned how to use arrays to store data. I practiced implementing linear and binary search methods. It was interesting to see how sorting the array made searching much faster. Understanding how to keep the array sorted while adding and removing artifacts was a key part of this task.

Difficulties and Solutions

Main Challenge: Ensuring that the array remained sorted after adding or removing items.
Solution: I implemented a method that inserts items in the correct position, maintaining the order necessary for binary search.

Further Improvements

- Add a feature to resize the array when it gets full.
- Enhance user-friendliness by allowing dynamic resizing of the ArtifactVault class.

---

2. The Linked List Labyrinth

What I Learned

In this challenge, I built a LabyrinthPath class using a singly linked list to model a path in a labyrinth. Each location was stored as a node, making it easy to add or remove locations. Implementing a loop detection algorithm was a valuable learning experience.

Difficulties and Solutions

Main Challenge: Detecting loops in the linked list.
Solution: I used Floyd's Cycle-Finding Algorithm, which provided a straightforward way to detect loops efficiently.

Further Improvements

- Add features to find the shortest path through the labyrinth.
- Allow saving and loading paths for future sessions.

---

3. The Stack of Ancient Texts

What I Learned

This challenge taught me about the stack data structure, which operates on a Last-In-First-Out (LIFO) basis. I implemented basic stack operations such as push, pop, and peek, helping me understand how stacks are used in applications like undo operations in software.

Difficulties and Solutions

Main Challenge: Handling stack underflow when popping from an empty stack.
Solution: I added checks to ensure that the stack wasn't empty before attempting to pop an item, which prevented errors and made the program more robust.

Further Improvements

- Implement a search method for specific scrolls.
- Enhance the functionality and usability of the ScrollStack class.

---

4. The Queue of Explorers

What I Learned

Through this challenge, I learned how circular queues work and why they are useful for managing data efficiently. I implemented methods to enqueue and dequeue explorers, which helped me understand the importance of maintaining a circular structure to save space.

Difficulties and Solutions

Main Challenge: Managing indices in a circular queue.
Solution: I used the modulo operator to ensure that the indices stayed within the bounds of the queue, preventing errors in indexing.

Further Improvements

- Implement a priority queue to allow some explorers to jump ahead in line based on certain criteria.
- Add extra complexity and functionality to the ExplorerQueue.

---

5. The Binary Tree of Clues

What I Learned

In this final challenge, I learned about binary trees and their traversal methods, such as in-order, pre-order, and post-order traversal. Implementing a binary tree helped me understand how data can be organized hierarchically.

Difficulties and Solutions

Main Challenge: Ensuring that the tree remained balanced.
Solution: I focused on how to properly insert nodes while keeping track of the tree structure to maintain balance.

Further Improvements

- Implement a self-balancing tree (e.g., AVL tree) to improve performance for insertions and deletions.
- Enhance searching speed for clues, especially as the number of clues increases.