Reflection on Challenges

1. Artifact Vault

The Artifact Vault challenge allowed me to practice sorting and searching algorithms. Implementing binary search on a sorted array of artifacts was a valuable exercise in efficiency. One difficulty I faced was keeping the vault sorted after each insertion and removal, but using the built-in sorting function simplified this. In the future, I could optimize this by implementing a custom insertion sort that avoids re-sorting the entire array every time an artifact is added or removed.

2. The Linked List Labyrinth

From this challenge, I learned the inner workings of linked lists and how they can be used to model real-world scenarios like a labyrinth path. One key takeaway was the use of the slow and fast pointer technique for loop detection. Implementing the methods to add, remove, and detect loops in the list helped deepen my understanding of how pointers and node traversal work in singly linked lists. A challenge I encountered was handling edge cases such as removing nodes from an empty list, which I solved by adding appropriate condition checks.

3. The Scroll Stack

This challenge taught me the importance of stacks in scenarios where data needs to be processed in a last-in-first-out (LIFO) order. Implementing the push, pop, and peek methods was relatively straightforward, but managing edge cases like popping from an empty stack required extra thought. I also appreciated learning how to check for the existence of an item within a stack. An improvement could be adding an option to dynamically increase the stack size if it gets full.

4. The Queue of Explorers

Working on the circular queue challenge improved my understanding of how queues can be implemented using arrays. The most interesting part was managing the wrap-around behavior when the queue becomes full and starts over at the beginning. I initially struggled with off-by-one errors in tracking the front and rear indices, but using modulo arithmetic helped me overcome this issue. In

the future, I would consider using a dynamic array or a linked list to avoid the fixed size limitation.

5. The Binary Tree of Clues

This challenge solidified my understanding of binary trees and traversal algorithms. Inserting nodes into a binary search tree (BST) and implementing in-order, pre-order, and post-order traversals helped me see the versatility of trees in data storage. One challenge was ensuring the tree remained balanced, though this was not strictly required. I also found the recursive nature of traversals challenging but rewarding once I mastered it. Future improvements could involve adding self-balancing techniques like AVL or Red-Black Trees.