Reflection on Assignment 1: Data Structures Adventure

Challenge 1: The Array Artifact

Reflection: In this challenge, I learned how to manage a fixed-size array without using built-in methods, which gave me a better understanding of memory management and manual implementation of search algorithms. Implementing both linear and binary search was insightful because it made me consider the trade-offs between efficiency and simplicity.

Difficulties Encountered: One of the challenges I faced was ensuring the binary search only works on a sorted array, so I had to manually verify the sorted order. I overcame this by adding logic to keep the array sorted while adding artifacts, which I later realized could be an extension to the current solution.

Ideas for Improvement: In the future, I could implement dynamic resizing of the array to avoid capacity limitations, or optimize the insertion method to always keep the array sorted for efficient binary searches.

Challenge 2: The Linked List Labyrinth

Reflection: This challenge reinforced my understanding of linked lists, especially in implementing node structures and managing references between them. The loop detection part was particularly rewarding as I explored Floyd's cycle-finding algorithm, which was a new concept for me.

Difficulties Encountered: Managing references when removing locations proved tricky, as I had to ensure there were no memory leaks or dangling references. After debugging several cases, I got a better grasp on handling linked list edge cases, such as removing the head node.

Ideas for Improvement: To improve this solution, I could introduce a feature to track the total distance traveled in the labyrinth or visualize the path using graphics.

Challenge 3: The Stack of Ancient Texts

Reflection: Implementing a stack manually without using built-in methods was a great way to practice fundamental operations like push, pop, and peek. It highlighted the LIFO (Last In, First Out) principle, which is essential for various applications in real-world problems.

Difficulties Encountered: The main difficulty was ensuring that stack overflow and underflow conditions were handled correctly. I overcame this by explicitly checking for these conditions before performing push or pop operations.

Ideas for Improvement: In the future, I could extend this stack to support undo/redo functionality, which would involve maintaining two stacks to track actions.

Challenge 4: The Queue of Explorers

Reflection: This challenge helped me deepen my understanding of circular queues. Implementing the queue operations like enqueue, dequeue, and handling the circular nature of the array required careful indexing and modular arithmetic.

Difficulties Encountered: I initially struggled with correctly wrapping the indices when the queue was full or empty. After walking through multiple test cases and revisiting the concept of circular queues, I refined the logic to handle these cases more effectively.

Ideas for Improvement: One possible extension would be to make the queue dynamic, allowing it to grow in size as more explorers join. Additionally, I could visualize the queue to represent explorers waiting in line more intuitively.

Challenge 5: The Binary Tree of Clues

Reflection: The binary tree challenge was an excellent exercise in recursive thinking. Implementing traversal methods like in-order, pre-order, and post-order solidified my understanding of tree structures. This challenge also introduced me to the concept of efficient searching and organizing hierarchical data.

Difficulties Encountered: The main challenge was getting the traversal algorithms correct, especially ensuring the recursive function calls were properly defined. Debugging traversal output was also a bit tricky initially, but once I visualized the tree, it became clearer.

Ideas for Improvement: In future iterations, I would like to implement self-balancing trees, such as AVL or Red-Black Trees, to ensure the tree remains balanced for faster operations.