Student Name: Sindu Rajkumar
Student ID: C0903440

Reflection on Coding Challenges

1. ArtifactVault Challenge

Reflection:

During the ArtifactVault challenge, I discovered how crucial it is to effectively manage collections with data structures. I was able to comprehend arrays and the intricacy of dynamic data manipulation on a deeper level by using an array-based method. In addition, I gained knowledge on how to apply binary and linear search algorithms, which are fundamental ideas in computer science.

Challenges and Solutions:

One of the main challenges I had was controlling the array's size and making sure that artifacts were removed effectively without compromising the integrity of the remaining pieces. I solved this problem by developing a technique that, upon removing an artifact, moves the remaining ones to fill the array with gaps.

Future Improvements:

To extend this solution, I could implement a dynamic array that resizes when it reaches capacity. Additionally, I could explore using a more sophisticated data structure, like a linked list, to allow for more flexible artifact management.

2. LabyrinthPath Challenge

Reflection:

I learned how important it is to use linked lists to express dynamic paths from the LabyrinthPath challenge. Understanding how nodes may be linked and traversed is crucial for comprehending more complicated data structures like trees and graphs, which is what I learned by building a linked structure to manage the path locations.

Challenges and Solutions:

One of the challenges I encountered was putting the strategy to look for path loops into practice. I made use of Floyd's cycle-finding technique, which found loops by utilizing two pointers. Though it took some careful consideration, this method helped me solve linked structure problems more effectively.

Future Improvements:

For future improvements, I could add methods to allow for backward navigation through the labyrinth or implement a more complex data structure that incorporates features like weighted paths or junctions for more intricate labyrinth designs.

3. ScrollStack Challenge

Reflection:

In the ScrollStack challenge, I learned how stacks operate on the principle of Last In, First Out (LIFO). This challenge emphasized the importance of stack operations and their applications in scenarios such as function calls and undo mechanisms in software.

Challenges and Solutions:

It was easy to implement the stack feature, although at first I had trouble keeping the stack in an empty state when doing pop and peek operations. I fixed this by making sure that appropriate condition checks were done before carrying out operations, which increased my stack implementation's dependability.

Future Improvements:

To enhance the ScrollStack class, I could implement a resizing mechanism to allow for a dynamic stack size. Additionally, I could include features such as tracking the history of popped items or implementing multiple stacks within a single class for better organization.

4. ExplorerQueue Challenge

Reflection:

The ExplorerQueue challenge provided me with a solid understanding of how queues work, particularly the First In, First Out (FIFO) principle. I learned how to manage queue operations effectively and the importance of ensuring that operations like enqueue and dequeue are efficient.

Challenges and Solutions:

I faced difficulties in managing the circular nature of the queue, specifically with updating the front and rear indices. I overcame this by using modular arithmetic to wrap around the indices when they reached the end of the array, which ensured the queue could function correctly without wasting space.

Future Improvements:

To improve the ExplorerQueue class, I could implement a dynamic resizing feature to expand or shrink the queue based on usage. Additionally, adding features like priority queuing could make the class more versatile in handling different types of explorer tasks.

5. ClueTree Challenge

Reflection:

The ClueTree challenge allowed me to explore binary trees and the operations associated with them. I gained experience in implementing traversal methods and understanding how trees can be used to organize data hierarchically, which is valuable for various applications.

Challenges and Solutions:

One challenge was ensuring that the insert operation maintained the properties of the binary search tree. I addressed this by thoroughly testing the insert and traversal methods to ensure they behaved as expected and maintained the correct structure.

Future Improvements:

To further enhance the ClueTree class, I could implement balancing techniques to ensure that the tree remains efficient as it grows. Additionally, implementing features to visualize the tree structure or supporting more complex queries could significantly improve its usability.