Reflection on Challenges

1. The Array Artifact

This challenge helped me to gather useful experience in the practical usage of arrays as an object storage and a tool for object retrieval. Thanks to this task I revisited array indexing and search algorithms and understood the difference between linear and binary search. In particular, binary search revealed conditions for the use of an array, namely, that it must be sorted for the algorithm to work properly.

Difficulty: These were main issues that were faced and the first question that needed an answer was how to maintain the array sorted during the insertion of new artifacts. To this, I countered by sorting the array after every insert and while this was effective, there should be more optimal ways of doing it.

Improvement Idea: An improvement might be to introduce an automatic sorting procedure that occurs on an instant basis at the time of insertion. This would maintain order and make the performing of operations under binary search possible.

2. The Linked List Labyrinth

This task helped me understand operations that can be performed on linked list, including node addition, but also removing last node, as well as loop detection. Pointers were needed to be managed carefully as they were used while applied with the loop detection algorithm.

Difficulty: At the beginning, identifying loops was not easy, but I had to maneuver that by use of Floyd's cycle-finding algorithm, which can also be referred to as the 'tortoise and hare' algorithm.

Improvement Idea: To optimize it even more I could given the fact that this is a two way list use a doubly linked list in order to efficiently back track.

3. The Stack of Ancient Texts

Although applying of the stack was easy enough, this task helped to emphasize the LIFO (Last In First Out) structure of stack. The task mainly concerned general stack tasks, for instance, pushing, popping, and peeking, in addition to calling for stack sanitation.

Difficulty: This became a little hard when dealing with edge conditions such as stack overflow or underflow but I was able to deal with this through checking, before any operation.

Improvement Idea: One idea is to build further on the concept of the stack and make it a dynamic structure that adaptively changes its size depending on the amount of new readings.

4. The Queue of Explorers

This enabled me to put into practice the use of circular queue in managing simple FIFO operations on this challenge. The exercise was to manage the front and the rear pointers particularly on the beginning of the queue position.

Difficulty: There were certain challenges in implementing the circular queue of which correct wrapping of the front and rear pointers at the time of enqueue and dequeue operations was moderately challenging.

Improvement Idea: This is true and incorporating modicum of dynamic resizing mechanism for the queue will make the queue more responsive to the current volumes of the data.

5. The Binary Tree of Clues

This challenge also increased my knowledge of binary search trees and the methods of insertion and traversal in recursion. I learnt how tree traversal techniques like in-order and post-in-order work and whether the node is visited before or after it is processed.

Difficulty: Balancing the tree while insertion and also while traversing was well understood by using recursion in the proper manner.

Improvement Idea: When dealing with rather skewed data – self Balancing or trees like AVL or Red black trees should be used to improve the efficiency.