**Day 1: Trees**

* **Workout 1: Insert into a Binary Tree**
  + Insert into a Binary Search Tree: <https://leetcode.com/problems/insert-into-a-binary-search-tree/>: Classic BST insertion problem.
* **Workout 2: Delete from a Binary Tree**
  + Delete Node in a BST: <https://leetcode.com/problems/delete-node-in-a-bst/> : Tests your understanding of BST manipulation.
* **Workout 3: Tree Traversal**
  + Binary Tree Inorder Traversal: <https://leetcode.com/problems/binary-tree-inorder-traversal/> : Core traversal algorithm, know all three (preorder, postorder, inorder).

**Day 2: BSTs**

* **Workout 1: Find Closest Value in BST**
  + Closest Binary Search Tree Value: <https://leetcode.com/problems/closest-binary-search-tree-value/>: Leverages the BST properties for efficient search.
* **Workout 2: Validate If a Tree is a BST**
  + Validate Binary Search Tree: <https://leetcode.com/problems/validate-binary-search-tree/> : A key algorithm in understanding BSTs.

**Day 3: Heaps**

* **Workout 1: Implement Heapsort**
  + While not asked directly, knowing the process greatly enhances your understanding of heaps. Look up how to implement it on LeetCode discussions or external resources.

**Day 4: Tries**

* **Workout 1: Find All Words with a Given Prefix**
  + Implement Trie (Prefix Tree): <https://leetcode.com/problems/implement-trie-prefix-tree/> (search function will be the focus here.)
* **Workout 2: Implement Autocomplete Suggestion**
  + This is more open-ended; start with a basic version by extending your Trie from Workout 1.

**Day 5: Graphs**

* **Workout 1: Find if a Path Exists Between Two Nodes**
  + Number of Islands: <https://leetcode.com/problems/number-of-islands/> (Though 'Medium', the core is pathfinding using DFS/BFS )
* **Workout 2: Find the Shortest Path Between Two Nodes**
  + Word Ladder: <https://leetcode.com/problems/word-ladder/> (A classic shortest path, usually solved with BFS)