**Week 1**

**Day 1: Array**

* Study Theory (1 hour)
  + Basic Operations: Insertion, deletion, searching.
  + Simple Manipulations: Reversal, rotation.
* Practice (1 hour)
  + Implement basic array operations: insertion, deletion, searching.
  + Solve simple array manipulation problems.

**Day 2: String**

* Study Theory (1 hour)
  + Basic Operations: Concatenation, substring search.
  + Pattern Matching: Brute force, naive pattern matching.
* Practice (1 hour)
  + Implement string operations: concatenation, substring search.
  + Solve problems related to pattern matching.

**Day 3: Hash Table**

* Study Theory (1 hour)
  + Basic Operations: Insertion, deletion, searching.
  + Collision Handling: Chaining, separate chaining.
* Practice (1 hour)
  + Implement basic hash table operations: insertion, deletion, searching.
  + Solve problems related to hash table collision handling.

**Day 4: Recursion**

* Study Theory (1 hour)
  + Understanding Recursion: Basic recursive functions (factorial, Fibonacci).
  + Recursive Backtracking: Simple problems like generating subsets, permutations.
* Practice (1 hour)
  + Implement recursive algorithms: factorial, Fibonacci.
  + Solve problems involving recursive backtracking.

**Day 5: Array (continued)**

* Study Theory (1 hour)
  + Searching Algorithms: Linear search, binary search.
  + Sorting Algorithms: Bubble sort, selection sort.
* Practice (1 hour)
  + Implement searching algorithms: linear search, binary search.
  + Implement sorting algorithms: bubble sort, selection sort.

**Day 6: String (continued)**

* Study Theory (1 hour)
  + String Reversal and Palindrome Detection: Basic approaches.
  + Regular Expressions: Basic syntax, simple pattern matching.
* Practice (1 hour)
  + Implement string reversal and palindrome detection algorithms.
  + Solve problems related to regular expressions.

**Day 7: Hash Table (continued)**

* Study Theory (1 hour)
  + Linear Probing: Handling collisions using linear probing.
  + Quadratic Probing: Handling collisions using quadratic probing.
* Practice (1 hour)
  + Implement hash table collision handling using linear probing.
  + Implement hash table collision handling using quadratic probing.

**Week 2**

**Day 1: Sorting and Searching**

* Study Theory (1 hour)
  + Sorting Algorithms: Insertion sort, selection sort.
  + Searching Techniques: Linear search, basic binary search.
* Practice (2 hours)
  + Implement insertion sort, selection sort.
  + Practice linear search, basic binary search.

**Day 2: Matrix**

* Study Theory (0.5 hours)
  + Matrix Operations: Addition, subtraction.
  + Basic Matrix Multiplication: Naive matrix multiplication.
* Practice (0.5 hours)
  + Implement addition, subtraction of matrices.
  + Implement naive matrix multiplication.

**Day 3: Linked List**

* Study Theory (1 hour)
  + Singly Linked List: Basic operations (insertion, deletion).
  + Doubly Linked List: Implementing doubly linked list operations.
* Practice (2 hours)
  + Implement basic operations of singly linked list.
  + Implement doubly linked list operations.

**Day 4: Queue**

* Study Theory (0.5 hours)
  + Queue Operations: Enqueue, dequeue.
  + Queue Implementation: Using arrays, linked lists.
* Practice (0.5 hours)
  + Implement enqueue, dequeue operations using arrays.
  + Implement enqueue, dequeue operations using linked lists.

**Day 5: Stack**

* Study Theory (0.5 hours)
  + Basic Stack Operations: Push, pop.
  + Stack Implementation: Using arrays, linked lists.
* Practice (0.5 hours)
  + Implement push, pop operations using arrays.
  + Implement push, pop operations using linked lists.

**Day 6: Queue (Continued)**

* Study Theory (0.5 hours)
  + Circular Queue: Implementing a circular queue.
  + Priority Queue: Basic priority queue implementation.
* Practice (0.5 hours)
  + Implement circular queue operations.
  + Implement basic priority queue operations.

**Day 7: Stack (Continued)**

* Study Theory (0.5 hours)
  + Deque: Implementing double-ended queues.
  + Applications: BFS traversal, level order traversal.
* Practice (0.5 hours)
  + Implement deque operations.
  + Solve problems related to BFS traversal and level order traversal.

**Week 3**

**Day 1: Tree**

* Study Theory (2 hours)
  + Binary Tree: Basic properties and operations.
  + Binary Search Tree (BST): Insertion, deletion, searching.
* Practice (2 hours)
  + Implement basic operations of binary tree.
  + Implement insertion, deletion, searching in BST.

**Day 2: Graph**

* Study Theory (2 hours)
  + Graph Representation: Adjacency matrix, adjacency list.
  + Graph Traversal: Depth-first search (DFS), breadth-first search (BFS).
* Practice (2 hours)
  + Implement graph representation using adjacency matrix and adjacency list.
  + Implement DFS and BFS traversal algorithms.

**Day 3: Heap**

* Study Theory (1.5 hours)
  + Min Heap and Max Heap: Understanding heap properties.
  + Heap Operations: Insertion, deletion, heapify.
* Practice (1.5 hours)
  + Implement basic heap operations: insertion, deletion, heapify.
  + Implement heap sort algorithm.

**Day 4: Tree (Continued)**

* Study Theory (1 hour)
  + AVL Tree: Understanding and balancing AVL trees.
  + Binary Heap: Basic operations (insertion, deletion).
* Practice (1 hour)
  + Implement AVL tree operations: balancing, rotation.
  + Implement basic operations of binary heap.

**Day 5: Graph (Continued)**

* Study Theory (1.5 hours)
  + Shortest Path Algorithms: Dijkstra's algorithm, Bellman-Ford algorithm.
  + Minimum Spanning Tree: Prim's algorithm, Kruskal's algorithm.
* Practice (1.5 hours)
  + Implement Dijkstra's algorithm for finding shortest paths.
  + Implement Prim's and Kruskal's algorithms for finding minimum spanning tree.

**Day 6: Heap (Continued)**

* Study Theory (1.5 hours)
  + Priority Queue: Implementing priority queue using heaps.
  + Heap Applications: kth largest/smallest element, median finding.
* Practice (1.5 hours)
  + Implement priority queue using heaps.
  + Solve problems related to finding kth largest/smallest element and median using heaps.

**Day 7: Trie**

* Study Theory (2 hours)
  + Basic Trie: Implementing basic trie data structure.
  + Trie Operations: Insertion, deletion, searching.
* Practice (2 hours)
  + Implement basic trie operations: insertion, deletion, searching.
  + Solve problems related to trie data structure.

**Week 4**

**Day 1: Interval**

* Study Theory (1 hour)
  + Interval Tree: Understanding and implementing interval tree.
  + Interval Queries: Range minimum query, range maximum query.
* Practice (1 hour)
  + Implement interval tree data structure.
  + Solve problems involving interval queries.

**Day 2: Dynamic Programming**

* Study Theory (2 hours)
  + Basic Concepts: Understanding the principles of dynamic programming.
  + Memoization: Applying memoization to optimize recursive solutions.
* Practice (2 hours)
  + Implement memoization technique for recursive solutions.
  + Solve problems using memoization approach.

**Day 3: Binary**

* Study Theory (1 hour)
  + Binary Representation: Understanding binary representation of numbers.
  + Bitwise Operations: Bitwise AND, OR, XOR operations.
* Practice (1 hour)
  + Perform bitwise operations.
  + Solve problems involving binary representation.

**Day 4: Math**

* Study Theory (1 hour)
  + Number Theory: Prime numbers, divisibility rules.
  + Modular Arithmetic: Solving problems using modular arithmetic.
* Practice (1 hour)
  + Solve problems involving prime numbers and divisibility rules.
  + Apply modular arithmetic to problem-solving.

**Day 5: Geometry**

* Study Theory (1 hour)
  + Basic Geometric Shapes: Points, lines, angles.
  + Coordinate Geometry: Understanding coordinate systems.
* Practice (1 hour)
  + Calculate distances and midpoints between points.
  + Solve problems involving equations of lines.

**Day 6: Interval (Continued)**

* Study Theory (1 hour)
  + Interval Intersection: Finding intersections between intervals.
  + Interval Union: Merging overlapping intervals.
* Practice (1 hour)
  + Implement algorithms for finding interval intersections and unions.
  + Solve problems involving interval scheduling.

**Day 7: Dynamic Programming (Continued)**

* Study Theory (1 hour)
  + Tabulation: Implementing bottom-up dynamic programming solutions.
  + Classic Problems: Longest common subsequence, knapsack problem.
* Practice (1 hour)
  + Implement tabulation approach for dynamic programming.
  + Solve problems related to classic dynamic programming problems.