100 Programming Exercises

Difficulty Distribution:

- **Easy (20):** Basic syntax, loops, conditionals
- **Medium (50):** Algorithms, data structures, problem-solving

- **Hard (20):** Complex algorithms, optimization
- **Very Hard (10):** Competition-level problems

🟢 EASY (20 Problems)

1. Simple Pattern

Problem: Print a right triangle pattern of stars with height N.

```
Input: 4
Output:
*
**
***
****
```

2. Sum of Digits

Problem: Calculate the sum of digits of a positive integer.

```
Input: 1234
Output: 10
```

3. Number Reversal

Problem: Reverse the digits of a positive integer.

```
Input: 12345
Output: 54321
```

4. Factorial Calculator

Problem: Calculate factorial of a non-negative integer.

```
Input: 5
Output: 120
```

5. Prime Check

Problem: Check if a number is prime.

```
Input: 17
Output: True
Input: 15
Output: False
```

6. Fibonacci Sequence

Problem: Print first N Fibonacci numbers.

```
Input: 7
Output: 0 1 1 2 3 5 8
```

7. Multiplication Table

Problem: Print multiplication table for a number up to 10.

```
Input: 5
Output: 5x1=5, 5x2=10, ..., 5x10=50
```

8. Count Vowels

Problem: Count vowels in a string.

```
Input: "Hello World"
Output: 3
```

9. Palindrome Check

Problem: Check if a string is palindrome.

```
Input: "radar"
Output: True
Input: "hello"
Output: False
```

10. Greatest Common Divisor

Problem: Find GCD of two numbers.

```
Input: 54, 24
Output: 6
```

11. Least Common Multiple

Problem: Find LCM of two numbers.

```
Input: 12, 18
Output: 36
```

12. Simple Calculator

Problem: Create a calculator for +, -, *, / operations.

```
Input: 5, 3, +
Output: 8
```

13. Temperature Converter

Problem: Convert Celsius to Fahrenheit and vice versa.

```
Input: 100C
Output: 212F
Input: 32F
Output: 0C
```

14. Odd/Even Counter

Problem: Count odd and even numbers in a list.

```
Input: [1,2,3,4,5,6,7,8,9]
Output: Odd:5, Even:4
```

15. List Maximum

Problem: Find maximum element in a list without built-in functions.

```
Input: [3,7,2,8,1,9,4]
Output: 9
```

16. String Reversal

Problem: Reverse a string without using built-in reverse.

```
Input: "python"
Output: "nohtyp"
```

17. Character Frequency

Problem: Count frequency of each character in a string.

```
Input: "programming"
Output: p:1, r:2, o:1, g:2, a:1, m:2, i:1, n:1
```

18. Simple Encryption

Problem: Shift each character in string by N positions in alphabet.

```
Input: "abc", 1
Output: "bcd"
```

19. Number to Words

Problem: Convert numbers 0-9 to words.

```
Input: 7
Output: "seven"
```

20. Area Calculator

Problem: Calculate area of circle, rectangle, triangle based on user choice.

```
Input: circle, radius=5
Output: 78.54
```

🟡 MEDIUM (50 Problems)

21. Matrix Multiplication

Problem: Multiply two matrices.

```
Input: [[1,2],[3,4]], [[5,6],[7,8]]
Output: [[19,22],[43,50]]
```

22. Binary Search

Problem: Implement binary search on sorted list.

```
Input: [1,3,5,7,9,11], target=7
Output: 3 (index)
```

23. Bubble Sort

Problem: Implement bubble sort algorithm.

```
Input: [64,34,25,12,22,11,90]
Output: [11,12,22,25,34,64,90]
```

24. Insertion Sort

Problem: Implement insertion sort algorithm.

```
Input: [12,11,13,5,6]
Output: [5,6,11,12,13]
```

25. Selection Sort

Problem: Implement selection sort algorithm.

```
Input: [29,10,14,37,13]
Output: [10,13,14,29,37]
```

26. Merge Sort

Problem: Implement merge sort algorithm.

```
Input: [38,27,43,3,9,82,10]
Output: [3,9,10,27,38,43,82]
```

27. Quick Sort

Problem: Implement quick sort algorithm.

```
Input: [10,7,8,9,1,5]
Output: [1,5,7,8,9,10]
```

28. Stack Implementation

Problem: Implement stack with push, pop, peek operations.

```
Input: push(1), push(2), pop(), push(3)
Output: [1,3] (stack state)
```

29. Queue Implementation

Problem: Implement queue with enqueue, dequeue operations.

```
Input: enqueue(1), enqueue(2), dequeue(), enqueue(3)
Output: [2,3] (queue state)
```

30. Linked List

Problem: Implement singly linked list with insert, delete, search.

```
Input: insert(1), insert(2), insert(3), delete(2)
Output: 1->3
```

31. Binary Tree Traversal

Problem: Implement inorder, preorder, postorder traversal.

```
Input: Binary tree with nodes 1,2,3,4,5
Output: Inorder: 4,2,5,1,3
```

32. Graph Representation

Problem: Create adjacency list and matrix for a graph.

```
Input: edges: (0,1), (0,2), (1,2), (2,3)
Output: Adjacency list representation
```

33. Depth First Search

Problem: Implement DFS for graph traversal.

```
Input: Graph with nodes 0,1,2,3, edges: (0,1),(0,2),(1,2),(2,3)
Output: DFS from 0: 0,1,2,3
```

34. Breadth First Search

Problem: Implement BFS for graph traversal.

```
Input: Same graph as above
Output: BFS from 0: 0,1,2,3
```

35. Dijkstra’s Algorithm

Problem: Find shortest path in weighted graph.

```
Input: Graph with weighted edges
Output: Shortest path distances from source
```

36. Knapsack Problem

Problem: Solve 0/1 knapsack problem.

```
Input: weights=[2,3,4,5], values=[3,4,5,6], capacity=5
Output: Maximum value: 7
```

37. Longest Common Subsequence

Problem: Find LCS of two strings.

```
Input: "ABCDGH", "AEDFHR"
Output: "ADH" (length 3)
```

38. Coin Change Problem

Problem: Find number of ways to make change for amount.

```
Input: coins=[1,2,5], amount=5
Output: 4 ways
```

39. Tower of Hanoi

Problem: Solve Tower of Hanoi with N disks.

```
Input: 3 disks
Output: Move sequences
```

40. N-Queens Problem

Problem: Place N queens on N×N chessboard without attacking.

```
Input: 4
Output: 2 solutions
```

41. Sudoku Solver

Problem: Solve 9×9 Sudoku puzzle.

```
Input: Partial Sudoku grid
Output: Solved grid
```

42. Expression Evaluation

Problem: Evaluate arithmetic expression with +,-,*,/,().

```
Input: "3 + (2 * 4)"
Output: 11
```

43. Infix to Postfix

Problem: Convert infix expression to postfix notation.

```
Input: "A+B*(C^D-E)"
Output: "ABCD^E-*+"
```

44. Postfix Evaluation

Problem: Evaluate postfix expression.

```
Input: "231*+9-"
Output: -4
```

45. Binary Search Tree

Problem: Implement BST with insert, search, delete operations.

```
Input: Insert 5,3,7,2,4,6,8
Output: Inorder traversal gives sorted list
```

46. AVL Tree

Problem: Implement self-balancing AVL tree.

```
Input: Insert sequence that causes rotations
Output: Balanced tree after each insertion
```

47. Hash Table

Problem: Implement hash table with chaining.

```
Input: Insert key-value pairs, handle collisions
Output: Correct retrieval after collisions
```

48. Priority Queue

Problem: Implement min-heap/max-heap.

```
Input: Insert elements with priorities
Output: Extract elements in priority order
```

49. Trie Data Structure

Problem: Implement trie for word storage and search.

```
Input: Insert "cat", "car", "card"
Output: Search "car" returns True, "cap" returns False
```

50. Union-Find Data Structure

Problem: Implement union-find with path compression.

```
Input: Union operations, find operations
Output: Correct connected components
```

51. Kruskal's Algorithm

Problem: Find MST using Kruskal's algorithm.

```
Input: Weighted graph
Output: Minimum spanning tree edges
```

52. Prim's Algorithm

Problem: Find MST using Prim's algorithm.

```
Input: Weighted graph
Output: Minimum spanning tree edges
```

53. Topological Sort

Problem: Perform topological sort on DAG.

```
Input: Directed acyclic graph
Output: Topological ordering
```

54. Strongly Connected Components

Problem: Find SCCs using Kosaraju's algorithm.

```
Input: Directed graph
Output: List of SCCs
```

55. Bellman-Ford Algorithm

Problem: Find shortest paths with negative weights.

```
Input: Graph with possible negative weights
Output: Shortest path distances or detect negative cycle
```

56. Floyd-Warshall Algorithm

Problem: Find all pairs shortest paths.

```
Input: Weighted graph
Output: Distance matrix between all pairs
```

57. Edit Distance

Problem: Calculate Levenshtein distance between strings.

```
Input: "kitten", "sitting"
Output: 3
```

58. Maximum Subarray Sum

Problem: Find contiguous subarray with maximum sum (Kadane’s).

```
Input: [-2,1,-3,4,-1,2,1,-5,4]
Output: 6 ([4,-1,2,1])
```

59. Longest Increasing Subsequence

Problem: Find length of longest increasing subsequence.

```
Input: [10,9,2,5,3,7,101,18]
Output: 4 ([2,3,7,101])
```

60. Matrix Chain Multiplication

Problem: Find optimal parenthesization for matrix multiplication.

```
Input: [10,30,5,60]
Output: Optimal cost: 4500
```

61. Rod Cutting Problem

Problem: Maximize profit from cutting rod.

```
Input: prices=[1,5,8,9,10,17,17,20], rod length=4
Output: Maximum profit: 10
```

62. Egg Dropping Puzzle

Problem: Find minimum trials to find critical floor.

```
Input: eggs=2, floors=10
Output: 4 trials
```

63. Josephus Problem

Problem: Find safe position in circle elimination.

```
Input: n=7, k=3
Output: Safe position: 3
```

64. Word Break Problem

Problem: Check if string can be segmented into dictionary words.

```
Input: "applepenapple", dictionary=["apple","pen"]
Output: True
```

65. Palindrome Partitioning

Problem: Partition string into palindrome substrings.

```
Input: "aab"
Output: [["a","a","b"],["aa","b"]]
```

66. Subset Sum Problem

Problem: Check if subset with given sum exists.

```
Input: [3,34,4,12,5,2], sum=9
Output: True
```

67. Rat in a Maze

Problem: Find all paths from top-left to bottom-right in maze.

```
Input: Maze with obstacles
Output: All possible paths
```

68. Tic-Tac-Toe AI

Problem: Implement minimax algorithm for Tic-Tac-Toe.

```
Input: Game state
Output: Best move for AI
```

69. Text Justification

Problem: Justify text to fit line width.

```
Input: words=["This","is","an","example"], width=16
Output: Formatted lines
```

70. Calculator with Memory

Problem: Implement calculator with memory functions (M+, M-, MR, MC).

```
Input: Sequence of operations
Output: Correct results with memory operations
```

🔴 HARD (20 Problems)

71. Traveling Salesman Problem

Problem: Solve TSP using dynamic programming (Held-Karp).

```
Input: Distance matrix between cities
Output: Minimum cost tour
```

72. Vertex Cover Problem

Problem: Find minimum vertex cover in graph.

```
Input: Graph
Output: Minimum vertex cover set
```

73. Hamiltonian Path/Cycle

Problem: Check if Hamiltonian path/cycle exists.

```
Input: Graph
Output: True/False and path if exists
```

74. Graph Coloring

Problem: Color graph with minimum colors (chromatic number).

```
Input: Graph
Output: Valid coloring with minimum colors
```

75. Maximum Flow (Ford-Fulkerson)

Problem: Find maximum flow in flow network.

```
Input: Flow network with capacities
Output: Maximum flow value
```

76. Minimum Cut Problem

Problem: Find minimum cut in flow network.

```
Input: Flow network
Output: Minimum cut edges and value
```

77. Bipartite Matching

Problem: Find maximum matching in bipartite graph.

```
Input: Bipartite graph
Output: Maximum matching
```

78. Segment Tree

Problem: Implement segment tree for range queries and updates.

```
Input: Array and range queries
Output: Efficient range query results
```

79. Fenwick Tree

Problem: Implement Fenwick tree for prefix sum queries.

```
Input: Array and prefix sum queries
Output: Efficient prefix sum calculations
```

80. Suffix Array

Problem: Build suffix array for string.

```
Input: String
Output: Suffix array
```

81. Longest Common Substring

Problem: Find longest common substring of multiple strings.

```
Input: Strings ["ABABC", "BABCA", "ABCBA"]
Output: "ABC" (length 3)
```

82. Burrows-Wheeler Transform

Problem: Implement BWT for data compression.

```
Input: String "banana"
Output: BWT transformed string
```

83. A* Search Algorithm

Problem: Implement A* pathfinding algorithm.

```
Input: Grid with obstacles, start, goal
Output: Optimal path
```

84. Monte Carlo Tree Search

Problem: Implement MCTS for game playing.

```
Input: Game state
Output: Best move after simulations
```

85. Neural Network from Scratch

Problem: Implement simple neural network with backpropagation.

```
Input: Training data
Output: Trained network for classification
```

86. Convex Hull (Graham Scan)

Problem: Find convex hull of points.

```
Input: Set of points
Output: Convex hull vertices in order
```

87. Closest Pair of Points

Problem: Find closest pair of points in O(n log n).

```
Input: Set of points
Output: Closest pair and distance
```

88. Line Segment Intersection

Problem: Find all intersections among line segments.

```
Input: Set of line segments
Output: All intersection points
```

89. Range Minimum Query

Problem: Answer RMQ efficiently with sparse tables.

```
Input: Array and multiple RMQs
Output: Minimum values for each query
```

90. Persistent Data Structures

Problem: Implement persistent segment tree.

```
Input: Sequence of updates and queries
Output: Correct results for each version
```

🏆 VERY HARD (10 Problems)

91. Fast Fourier Transform

Problem: Implement FFT for polynomial multiplication.

```
Input: Two polynomials as coefficient arrays
Output: Their product polynomial
```

92. Suffix Automaton

Problem: Build suffix automaton for string.

```
Input: String
Output: Suffix automaton structure
```

93. Heavy-Light Decomposition

Problem: Implement HLD for tree path queries.

```
Input: Tree and path queries
Output: Efficient query results
```

94. Link-Cut Trees

Problem: Implement dynamic trees with link/cut operations.

```
Input: Dynamic tree operations
Output: Correct tree state after operations
```

95. Minimum Cost Maximum Flow

Problem: Solve min-cost max-flow problem.

```
Input: Flow network with costs
Output: Maximum flow with minimum cost
```

96. Simplex Algorithm

Problem: Implement simplex method for linear programming.

```
Input: Linear program in standard form
Output: Optimal solution
```

97. Hungarian Algorithm

Problem: Solve assignment problem.

```
Input: Cost matrix
Output: Minimum cost assignment
```

98. Miller-Rabin Primality Test

Problem: Implement probabilistic primality test.

```
Input: Large number
Output: Probabilistic prime determination
```

99. Pollard’s Rho Algorithm

Problem: Implement integer factorization algorithm.

```
Input: Composite number
Output: One non-trivial factor
```

100. Aho-Corasick Algorithm

Problem: Implement multiple pattern matching.

```
Input: Text and multiple patterns
Output: All occurrences of all patterns
```

📝 Problem Structure Template (for reference):

Each problem should clearly specify: - **Input Format:** Data types, constraints, examples
- **Output Format:** Expected result format
- **Constraints:** Time/space limits, input ranges - **Sample Input/Output:** Multiple test cases - **Explanation:** Brief reasoning for sample cases

🎯 Assessment Focus:

- **Easy:** Basic syntax mastery, simple algorithms
- **Medium:** Algorithm implementation, data structure usage
- **Hard:** Complex algorithm design, optimization
- **Very Hard:** Advanced CS concepts, competition-level thinking