C Programs: - Beginner Level

Phase 1: The Basics (Input, Output, & Operators)

1. Print "Hello, World!".
2. Read an integer/float/char and print it.
3. Swap two numbers using a temporary variable.
4. **Swap two numbers *without* a temporary variable.**
5. Convert temperature from Celsius to Fahrenheit (and vice versa).
6. Calculate Simple Interest and Compound Interest.
7. Calculate the area and circumference of a circle.
8. Check if a number is even or odd (using modulus operator).
9. Check if a number is even or odd (using **bitwise** operators).
10. Find the size of different data types (int, char, float, double) using sizeof.
11. Convert days into years, weeks, and days.
12. Calculate the ASCII value of a character.

Phase 2: Control Flow (Conditionals & Loops)

Conditionals (If/Else/Switch)

13. Find the largest of three numbers.
14. Check if a year is a leap year.
15. Check if a character is a vowel or consonant.
16. Find the roots of a quadratic equation.
17. Create a simple calculator using switch case.
18. Check if a character is an alphabet, digit, or special character.

Looping Logic (For/While/Do-While)

19. Calculate the factorial of a number.
20. Generate the **Fibonacci sequence** up to N terms.
21. Check if a number is a **Prime number**.
22. Print all Prime numbers between 1 and N.
23. Check if a number is a **Palindrome** (e.g., 121).
24. Check if a number is an **Armstrong number** (e.g., 153).
25. Check if a number is a **Strong number** (sum of factorial of digits = number).

26. Reverse a given integer.
27. Calculate the sum of digits of a number.
28. Find the GCD (HCF) and LCM of two numbers.

Pattern Printing (Visual Logic)

29. Print a half-pyramid of stars (*).
30. Print an inverted half-pyramid.
31. Print a full pyramid (triangle).
32. Print a diamond shape.
33. Print **Pascal's Triangle**.
34. Print Floyd's Triangle.
35. Print a number pyramid (1, 12, 123, etc.).

Phase 3: Arrays and Matrix Operations

36. Find the sum and average of array elements.
37. Find the **largest and smallest** element in an array.
38. **Reverse** an array in place (without a second array).
39. Insert an element at a specific position in an array.
40. Delete an element from a specific position.
41. Remove duplicate elements from an array.
42. Merge two arrays.
43. Rotate an array left or right by N positions.
44. **Linear Search:** Find an element in an array.
45. **Binary Search:** Find an element in a sorted array.
46. **Bubble Sort:** Sort an array in ascending order.
47. **Selection Sort:** Sort an array.

2D Arrays (Matrices)

48. Add two matrices.
49. **Multiply two matrices** (Classic problem).
50. Find the Transpose of a matrix.
51. Check if a matrix is symmetric.
52. Calculate the sum of diagonal elements.

53. Check if two matrices are equal.

Phase 4: Strings

54. Find the length of a string without using strlen.
55. Copy one string to another without strcpy.
56. Concatenate two strings without strcat.
57. Compare two strings without strcmp.
58. Reverse a string.
59. Check if a string is a **Palindrome**.
60. Count the number of vowels, consonants, digits, and spaces in a string.
61. Convert a string to lowercase/uppercase.
62. Sort a set of strings in alphabetical order.
63. Remove all characters in a string except alphabets.
64. Find the frequency of characters in a string.

Phase 5: Pointers (The Heart of C)

65. Swap two numbers using pointers.
66. Add two numbers using pointers.
67. Input and print array elements using pointers.
68. Copy one array to another using pointers.
69. Swap two arrays using pointers.
70. Reverse an array using pointers.
71. Search for an element in an array using pointers.
72. Add two matrices using pointers.
73. **Pointer to Pointer:** Demonstrate the use of double pointers.

Phase 6: Functions & Recursion

74. Create a function to check for Prime/Armstrong/Perfect numbers.
75. **Call by Value vs. Call by Reference** demonstration.
76. Find the factorial of a number using **Recursion**.
77. Find the GCD of two numbers using Recursion.

78. Solve the **Tower of Hanoi** problem.
79. Reverse a sentence using Recursion.
80. Calculate the power of a number using Recursion.
81. Find the sum of natural numbers using Recursion.

Phase 7: Structures and Unions

82. Store information of a student (name, roll, marks) using Structure.
83. Add two distances (in inch-feet) system using Structures.
84. Add two Complex numbers using structures.
85. Calculate the difference between two time periods using structures.
86. Store information of 10 students using an **Array of Structures**.
87. Demonstrate the difference in memory usage between Structure and Union.

Phase 8: File Handling

88. Write a sentence to a file.
89. Read a file and display its content on the console.
90. Copy the content of one file to another.
91. Merge two files into a third file.
92. Count the number of lines and words in a text file.
93. Delete a specific line from a file.
94. Replace a specific word in a file.
95. Create a simple "Student Management System" using files (Create, Read, Update, Delete records).

Phase 9: Advanced / Data Structures

96. Implement a **Linked List** (Insert, Delete, Display).
97. Reverse a Linked List.
98. Implement a **Doubly Linked List**.
99. Implement a **Stack** using Arrays.
100. Implement a **Queue** using Arrays.
101. Implement a Stack using Pointers (Linked List).

102. Check for balanced parentheses using a Stack.

103. **Bit Manipulation:** Check if the -th bit is set.

104. **Bit Manipulation:** Toggle the -th bit.

105. Count set bits in an integer.

C Programs: - Intermediate & Advanced Level

Phase 1: Advanced Loops & Number Theory (40 Questions)

1. Check if a number is a **Perfect Number**.
2. Print all Perfect Numbers between 1 and 10,000.
3. Check if a number is a **Happy Number**.
4. Check if a number is a **Neon Number**.
5. Check if a number is a **Spy Number**.
6. Check if a number is an **Automorphic Number**.
7. Check if a number is a **Harshad (Niven) Number**.
8. Check if a number is a **Pronic Number**.
9. Print the **Collatz Conjecture** sequence for any number n.
10. Find the sum of the series: $1 + 1/2 + 1/3 + ... + 1/n$.
11. Find the sum of the series: $1 - 1/2 + 1/3 - 1/4 ...$.
12. Calculate the value of π using the Leibniz formula series.
13. Print Pascal's Triangle using a 2D array.
14. Convert a Binary number to Decimal.
15. Convert a Decimal number to Binary.
16. Convert a Decimal number to Octal.
17. Convert a Decimal number to Hexadecimal.
18. Convert Hexadecimal to Decimal.
19. Find the prime factors of a number.
20. Check if two numbers are **Amicable Numbers**.
21. Find the LCM of *n* numbers.
22. Find the GCD of *n* numbers.

23. Calculate x^y without using pow() function.
24. Calculate the square root of a number without sqrt() (Newton-Raphson method).
25. Find the generic root of a number (sum digits until single digit).
26. Print numbers in words (e.g., 123 -> "One Two Three").
27. Print numbers in words (e.g., 123 -> "One Hundred Twenty Three").
28. Generate the Lucas Sequence.
29. Find the number of trailing zeros in a factorial.
30. Find the last non-zero digit of a factorial.
31. Check if a number is a Mersenne Prime.
32. Find the sum of prime numbers in a given range.
33. Find the largest prime factor of a number.
34. Check if a number can be expressed as a sum of two primes (Goldbach's Conjecture).
35. Find the Catalan number for n.
36. Print a spiral pattern of numbers (Spiral Matrix).
37. Print a hollow square star pattern.
38. Print an "X" shape using stars.
39. Print a Butterfly pattern using stars.
40. Print a Zig-Zag pattern.

Phase 2: Advanced Arrays & Matrices (30 Questions)

41. Find the "Leader" elements in an array (elements greater than all elements to their right).
42. Find the majority element in an array (element appearing > N/2 times).
43. Find the missing number in an array of 1 to N.
44. Find the repeating number in an array of 1 to N.
45. Move all zeros to the end of an array.
46. Find the intersection of two arrays.
47. Find the union of two arrays.
48. Find the subarray with the maximum sum (**Kadane's Algorithm**).
49. Find the maximum product subarray.
50. Rearrange an array such that arr[i] = i.
51. Segregate even and odd numbers in an array.

52. Find the Kth largest element in an array.
53. Find the Kth smallest element in an array.
54. Find specific pair in array whose sum is X.
55. Find triplets in array whose sum is X.
56. Rotate a matrix 90 degrees clockwise.
57. Rotate a matrix 90 degrees anti-clockwise.
58. Print a matrix in a generic Spiral Order.
59. Sort elements of each row of a matrix.
60. Sort elements of each column of a matrix.
61. Find the saddle point of a matrix.
62. Check if a matrix is an Identity Matrix.
63. Check if a matrix is a Sparse Matrix.
64. Convert a Sparse Matrix into a condensed format (Tuple).
65. Find the determinant of a 2×2 matrix.
66. Find the determinant of a 3×3 matrix.
67. Find the inverse of a 2×2 matrix.
68. Multiply a matrix by a scalar.
69. Interchange the diagonals of a matrix.
70. Find the lower triangular matrix.

Phase 3: Deep Dive Strings (30 Questions)

71. Implement your own strstr() (Find substring index).
72. Implement your own atoi() (String to Integer).
73. Implement your own itoa() (Integer to String).
74. Check if two strings are **Anagrams**.
75. Check if a string is a **Pangram** (contains all alphabets).
76. Find the first non-repeating character in a string.
77. Remove duplicate characters from a string.
78. Find the longest palindrome substring.
79. Count the occurrences of a specific word in a string.
80. Capitalize the first letter of every word.

81. Swap the case of every character (Lower <-> Upper).
82. Remove all extra spaces from a string (trimming).
83. Sort characters of a string using Bubble Sort.
84. Find the lexicographically smallest rotation of a string.
85. Print all permutations of a string.
86. Print all subsequences of a string.
87. Check if a string is a valid email address (simple validation).
88. Compress a string using counts (e.g., "aaabb" -> "a3b2").
89. Expand a compressed string ("a3b2" -> "aaabb").
90. Reverse words in a given string (e.g., "Hello World" -> "World Hello").
91. Find the length of the longest word in a string.
92. Encode a string using Caesar Cipher.
93. Decode a Caesar Cipher string.
94. Check if a string has balanced brackets.
95. Find the Longest Common Prefix in an array of strings.
96. Check if one string is a rotation of another.
97. Count the number of special characters in a string.
98. Remove specific characters from a string.
99. Replace a substring with another string.
100. Convert a Roman Numeral to an Integer.

Phase 4: Bitwise Magic (25 Questions)

101. Check if a number is a power of 2 using bitwise.
102. Check if a number is a power of 4.
103. Swap two nibbles in a byte.
104. Reverse bits of an integer.
105. Find the only non-repeating element in an array where every other element repeats twice.
106. Find the two non-repeating elements in an array.
107. Count total set bits in all numbers from 1 to N.
108. Find the position of the only set bit.

109. Detect if two integers have opposite signs.
110. Add 1 to an integer using bitwise operators.
111. Multiply a number by 3.5 using bitwise.
112. Compute the absolute value without branching.
113. Find the maximum of two numbers without comparison operators.
114. Find the minimum of two numbers without comparison operators.
115. Swap odd and even bits.
116. Convert binary code to Gray code.
117. Convert Gray code to binary.
118. Find the XOR sum of all elements in an array.
119. Check if binary representation of a number is a palindrome.
120. Turn off the rightmost set bit.
121. Check if a number has alternating bits.
122. Find the Hamming Distance between two integers.
123. Calculate parity of a number.
124. Divide integers without division operator.
125. Implement a simple bitset.

Phase 5: Recursion & Backtracking (25 Questions)

126. Print numbers from 1 to N without loops (Recursion).
127. Print array elements using recursion.
128. Find the maximum element in an array using recursion.
129. Check if a string is palindrome using recursion.
130. Generate all binary strings of length N.
131. **N-Queens Problem**.
132. **Rat in a Maze Problem**.
133. **Sudoku Solver**.
134. Generate all subsets of a set.
135. Permutations of an array.
136. Combination Sum problem.
137. Flood Fill Algorithm (like Paint Bucket tool).

138. Josephus Problem.
139. Ackermann function implementation.
140. Find the number of paths in a grid from (0,0) to (n,m).
141. Reverse a Stack using recursion.
142. Sort a Stack using recursion.
143. Check if an array is sorted using recursion.
144. Print a Linked List in reverse using recursion.
145. Convert a string to integer using recursion.
146. Count the steps to solve Tower of Hanoi.
147. Remove duplicates from a string using recursion.
148. Find the modular exponentiation $(a^b) \% m$.
149. Knapsack Problem (0/1) using recursion.
150. String interleaving problem.

Phase 6: Data Structures (Implementation) (30 Questions)

151. Implement a **Circular Linked List**.
152. Detect a cycle in a Linked List (Floyd's Cycle Finding).
153. Find the starting node of the cycle in a Linked List.
154. Find the middle element of a Linked List.
155. Merge two sorted Linked Lists.
156. Remove the N-th node from the end of a Linked List.
157. Check if a Linked List is a palindrome.
158. Intersection point of two Linked Lists.
159. Implement a **Queue using two Stacks**.
160. Implement a **Stack using two Queues**.
161. Implement a Circular Queue using Array.
162. Implement a Priority Queue.
163. Implement a Deque (Double Ended Queue).
164. Implement a **Binary Search Tree (BST)** (Insert, Delete, Search).
165. Find the Height of a Binary Tree.
166. Tree Traversals: Inorder, Preorder, Postorder.

167. Level Order Traversal of a Binary Tree.
168. Check if two trees are identical.
169. Find the Lowest Common Ancestor (LCA) in a BST.
170. Check if a Binary Tree is a BST.
171. Find the diameter of a Binary Tree.
172. Convert a Binary Tree to its Mirror.
173. Count leaf nodes in a Binary Tree.
174. Check if a Binary Tree is balanced.
175. Implement a Min Heap.
176. Implement a Max Heap.
177. Heap Sort implementation.
178. Implement an Adjacency Matrix for a Graph.
179. Implement BFS (Breadth First Search) for a Graph.
180. Implement DFS (Depth First Search) for a Graph.

Phase 7: System & Preprocessor (20 Questions)

181. Define a Macro to find the square of a number.
182. Define a Macro to swap two numbers.
183. Use #ifdef, #ifndef, #else, #endif for conditional compilation.
184. Implement a Macro to print a variable name as a string (# operator).
185. Implement the token pasting operator (##).
186. Write a C program that prints its own source code.
187. Use setjmp and longjmp for exception handling.
188. Measure the execution time of a function using time.h.
189. Generate random numbers using rand() and srand().
190. Use realloc() to dynamically resize an array.
191. Create a memory leak intentionally and then fix it.
192. Handle command-line arguments (argc, argv).
193. Write a program to delete itself after execution.
194. Check Endianness of the system (Little Endian or Big Endian).
195. Implement a variable argument function (like printf) using stdarg.h.

196. Execute a system command (like ls or dir) from C program.
197. Get the current date and time.
198. Use enum to create a boolean type.
199. Use typedef to create function pointers.
200. Create a flexible array member in a structure.

Phase 8: Advanced Sorting & Searching (20 Questions)

201. Implement **Merge Sort** (Recursive).
202. Implement Merge Sort (Iterative).
203. Implement **Quick Sort** (Recursive).
204. Implement Quick Sort (Iterative using Stack).
205. Implement **Randomized Quick Sort**.
206. Implement **Counting Sort**.
207. Implement **Radix Sort**.
208. Implement **Bucket Sort**.
209. Implement **Shell Sort**.
210. Implement **Comb Sort**.
211. Implement **Cocktail Shaker Sort**.
212. Implement **Gnome Sort**.
213. Implement **Interpolation Search**.
214. Implement **Exponential Search**.
215. Implement **Ternary Search**.
216. Find the Kth smallest element using QuickSelect.
217. Sort an array of strings using Quick Sort.
218. Sort an array of structures based on a specific key.
219. Check if an array is sorted and rotated.
220. Find the peak element in an array (element greater than neighbors).

Phase 9: Graph Algorithms (30 Questions)

221. Represent a graph using an Adjacency List.
222. Count the number of edges in a graph.
223. **Dijkstra's Algorithm** for shortest path.

224. **Bellman-Ford Algorithm** (detect negative cycles).
225. **Floyd-Warshall Algorithm** (All-pairs shortest path).
226. **Prim's Algorithm** for Minimum Spanning Tree (MST).
227. **Kruskal's Algorithm** for MST (using Union-Find).
228. Detect a cycle in a Directed Graph.
229. Detect a cycle in an Undirected Graph.
230. **Topological Sort** of a Directed Acyclic Graph (DAG).
231. Check if a graph is Bipartite.
232. Find all Connected Components in a graph.
233. Find the number of islands in a grid (Graph logic on 2D array).
234. Find the shortest path in a Binary Maze.
235. Check if a path exists between two vertices.
236. Find Articulation Points in a graph.
237. Find Bridges in a graph.
238. Transitive Closure of a graph using DFS.
239. Check if a graph is a Tree.
240. Find the maximal clique size.
241. Implement graph coloring (Greedy).
242. Solve the Traveling Salesman Problem (Brute force/DP).
243. Find the strongly connected components (Kosaraju's Algorithm).
244. Find the strongly connected components (Tarjan's Algorithm).
245. Hierholzer's Algorithm for Eulerian Path.
246. Check if a graph consists of a Hamiltonian Cycle.
247. Clone a Graph.
248. Word Ladder Problem (BFS).
249. Maximum Flow Problem (Ford-Fulkerson).
250. Snake and Ladder Problem using BFS.

Phase 10: Dynamic Programming (DP) (30 Questions)

251. **Fibonacci Series** using Memoization (Top-down).
252. Fibonacci Series using Tabulation (Bottom-up).

253. **Longest Common Subsequence (LCS)**.
254. Longest Increasing Subsequence (LIS).
255. Longest Palindromic Subsequence.
256. Edit Distance (Levenshtein Distance).
257. 0/1 Knapsack Problem.
258. Unbounded Knapsack Problem.
259. Subset Sum Problem.
260. Partition Equal Subset Sum.
261. Coin Change Problem (Minimum coins).
262. Coin Change Problem (Number of ways).
263. Rod Cutting Problem.
264. Egg Dropping Puzzle.
265. Matrix Chain Multiplication.
266. Minimum Path Sum in a Grid.
267. Unique Paths in a Grid.
268. Minimum jumps to reach the end of an array.
269. House Robber Problem.
270. Maximum Sum Increasing Subsequence.
271. Count number of ways to cover a distance (Steps 1, 2, 3).
272. Maximum length chain of pairs.
273. Longest Common Substring.
274. Shortest Common Supersequence.
275. Count Palindromic Substrings.
276. Wildcard Pattern Matching.
277. Regular Expression Matching.
278. Burst Balloons Problem.
279. Palindrome Partitioning.
280. Largest Square of 1s in a Binary Matrix.

Phase 11: Advanced Data Structures (20 Questions)

281. Implement a **Trie (Prefix Tree)** (Insert, Search).

282. Delete a word from a Trie.
283. Implement an **AVL Tree** (Insertion with rotation).
284. Implement a **Red-Black Tree** (Insertion logic).
285. Implement a **Segment Tree** (Range Sum Query).
286. Implement a **Fenwick Tree (Binary Indexed Tree)**.
287. Find the Lowest Common Ancestor using Segment Tree.
288. Implement Disjoint Set (Union-Find) with path compression.
289. Serialize and Deserialize a Binary Tree.
290. Construct a Tree from Inorder and Preorder traversal.
291. Construct a Tree from Inorder and Postorder traversal.
292. Find all nodes at distance K from a target node.
293. Maximum Path Sum in a Binary Tree.
294. Flatten a Binary Tree to a Linked List.
295. Convert a BST to a Doubly Linked List.
296. Print the boundary traversal of a Binary Tree.
297. Print the view of a tree (Top, Bottom, Left, Right).
298. Check if a Binary Tree is a Subtree of another.
299. ZigZag Level Order Traversal.
300. Implement a Suffix Tree (Conceptual/Simple version).

Phase 12: System Programming & OS Concepts (30 Questions)

301. **Process Creation:** Use fork() to create a child process.
302. **Process ID:** Get PID and PPID of processes.
303. **Wait:** Use wait() to prevent zombie processes.
304. **Exec:** Use exec() family functions to run shell commands.
305. **Signals:** Write a signal handler for SIGINT (Ctrl+C).
306. **Pipes:** Implement Inter-Process Communication (IPC) using pipe().
307. **Named Pipes:** Implement IPC using FIFOs.
308. **Shared Memory:** Write/Read to shared memory segment.
309. **Threads:** Create threads using pthread_create.
310. **Mutex:** Solve a Race Condition using Mutex locks.

311. **Semaphores:** Implement the Producer-Consumer problem.
312. **Dining Philosophers Problem:** Simulation.
313. **Banker's Algorithm:** Deadlock avoidance simulation.
314. **Page Replacement:** Simulate FIFO Page Replacement.
315. **Page Replacement:** Simulate LRU (Least Recently Used) Replacement.
316. **Scheduling:** Simulate FCFS (First Come First Serve).
317. **Scheduling:** Simulate SJF (Shortest Job First).
318. **Scheduling:** Simulate Round Robin Scheduling.
319. Read Environment Variables using getenv.
320. Set Environment Variables using setenv.
321. Get current working directory (getcwd).
322. Change directory (chdir).
323. List files in a directory (dirent.h).
324. Check file permissions using access().
325. Get file metadata (size, time) using stat().
326. Create a daemon process.
327. Implement a simple Shell (command interpreter).
328. Catch Floating Point Exceptions (SIGFPE).
329. Memory mapping (mmap) a file.
330. Lock a file using fcntl.

Phase 13: Advanced Pointers & Memory (20 Questions)

331. Implement malloc(), calloc(), free() from scratch (simple version).
332. Detect Memory Leaks (conceptual wrapper).
333. Use **Function Pointers** to implement a callback system.
334. Implement a generic sorting function (like qsort) using void*.
335. Create an array of function pointers (Menu driven system).
336. Pass a 2D array to a function using pointer to an array.
337. Access a specific memory address (dangerous but educational).
338. Cast a pointer to an integer and back.
339. Implement a flexible buffer (dynamic resizing).

340. Use restrict keyword and demonstrate its effect.
341. Use volatile keyword in a multithreaded context.
342. Implement a Smart Pointer (using structure and ref counting).
343. Hex dump of a memory region.
344. Compare two memory blocks using memcmp.
345. Copy overlapping memory blocks using memmove vs memcpy.
346. Create a jagged array (dynamic 2D array with variable row lengths).
347. Pointer arithmetic on void* (casting required).
348. Return multiple values from a function using pointers.
349. Implement a simple Garbage Collector logic.
350. Flatten a multi-level linked list.

Phase 14: Parsing & String Algorithms (20 Questions)

351. Implement **KMP Algorithm** (Pattern Searching).
352. Implement **Rabin-Karp Algorithm**.
353. Implement **Boyer-Moore Algorithm**.
354. Write a JSON Parser (Simple Key-Value).
355. Write a CSV Parser.
356. Write an XML Parser (Basic tag matching).
357. Evaluate a mathematical expression (String) (e.g., "3+5*2").
358. Convert Infix expression to Postfix.
359. Convert Infix expression to Prefix.
360. Evaluate a Postfix expression.
361. Validate an IP Address (IPv4).
362. Compress a string using Huffman Coding.
363. Implement Run Length Encoding.
364. Find the longest repeated subsequence in a string.
365. Check if a string follows a given pattern (e.g., "abba" -> "dog cat cat dog").
366. Minimum window substring containing all characters of another string.
367. Group Anagrams together.
368. Basic HTML tag stripper.

369. URL Encode a string.

370. URL Decode a string.

Phase 15: Game Development (Console) (15 Questions)

371. **Tic-Tac-Toe** (Player vs Player).

372. Tic-Tac-Toe (Player vs Computer - Random).

373. Tic-Tac-Toe (Player vs Computer - Minimax Algorithm).

374. **Hangman** Game.

375. **Snake** Game (using kbhit or non-blocking input).

376. **Minesweeper** (Grid generation and logic).

377. **Sudoku** Validator.

378. **Sudoku** Generator.

379. **2048** Game Logic.

380. Rock, Paper, Scissors.

381. Maze generator (Randomized DFS).

382. Connect 4 Game.

383. Blackjack (Card game logic).

384. Number Guessing Game with Binary Search feedback (Higher/Lower).

385. Battleship Game (Grid logic).

Phase 16: Mathematical & Cryptography (15 Questions)

386. **Sieve of Eratosthenes** (Prime generation).

387. Segmented Sieve.

388. **Euclid's Extended Algorithm**.

389. Modular Multiplicative Inverse.

390. Chinese Remainder Theorem.

391. Matrix Exponentiation (Fast Fibonacci).

392. RSA Encryption (Simple implementation with small numbers).

393. Caesar Cipher (Shift).

394. Vigenère Cipher.

395. Check if points are collinear.

396. Find the Convex Hull (Jarvis March).

397. Find the area of a polygon.

398. Check if two line segments intersect.

399. Compute the hash of a string (Rolling Hash).

400. Generate a random password.