
Category 1 — Template & Metaprogramming Hybrid (10 problems)

1. Implement a **compile-time Fibonacci generator** and also compute it at runtime using the same function.
2. Build a **type-safe matrix class** that supports only arithmetic types, using templates and static_assert.
3. Implement a **constexpr factorial** function and verify correctness at compile-time.
4. Create a **template that detects if a type is iterable** at compile-time.
5. Implement a **static polymorphic container** that stores different types but enforces size constraints at compile-time.
6. Build a **template-based unit system** (meters, seconds) that allows safe compile-time dimensional arithmetic.
7. Implement a **compile-time type list** with push_back, contains, and remove operations.
8. Build a **template to compute largest element type in a type list** at compile-time.
9. Implement **template specialization to select optimal container type** for given data size.
10. Create a **template-based fixed-size vector** with operations only allowed at compile-time for constexpr inputs.

Category 2 — Algorithmic Challenges (10 problems)

11. Implement **convex hull algorithm** using std::vector and custom comparators.
12. Build a **dynamic connectivity checker** using union-find with path compression.
13. Implement **segment tree** supporting range sum and update.
14. Build **Fenwick tree / Binary Indexed Tree** for prefix sums.
15. Implement **sparse table** for range minimum queries.
16. Build a **disjoint set with rollback support** for persistent union-find.
17. Implement **top-K frequent elements** using STL heap algorithms and map.
18. Build a **weighted interval scheduling** algorithm using vectors and binary search.
19. Implement **offline LCA queries** on tree using Euler tour + RMQ.
20. Implement **2-SAT solver** using implication graph and DFS.

Category 3 — Advanced Pointer/Ownership (10 problems)

21. Implement **shared ownership of a dynamically allocated buffer** with manual reference counting.

22. Build a **linked list with shared nodes** using weak pointers to prevent leaks.
23. Implement **smart pointer with delayed destruction** for performance-sensitive objects.
24. Build a **circular linked list with safe traversal under deletion**.
25. Implement a **tree structure with multiple parents** using shared_ptr + weak_ptr safely.
26. Build a **pool of objects** where ownership is dynamically transferred between threads.
27. Implement **manual copy-on-write string** with custom allocation.
28. Build a **dynamic array of polymorphic objects** with unique_ptr ownership.
29. Implement **thread-safe reference-counted observer pattern**.
30. Build a **buffer chain** for network packets where each buffer owns next buffer partially.

Category 4 — Hybrid Container/Algorithm (10 problems)

31. Implement a **deque of deques** supporting efficient flattening into vector.
32. Build a **map of sets** and implement a function to compute union and intersection efficiently.
33. Implement **sliding window median** using two STL heaps.
34. Build a **2D interval tree** using vectors of vectors for range queries.
35. Implement **graph adjacency list with sorted neighbors** and efficient neighbor search.
36. Build **persistent vector** that supports snapshot and rollback.
37. Implement **matrix chain multiplication** optimizer using vectors and DP.
38. Build a **dynamic trie structure** supporting insert, search, and prefix search.
39. Implement **deque-based monotonic queue** for minimum/maximum in a window.
40. Build **mergeable heaps** using STL containers with logarithmic merges.

Category 5 — Advanced Functional-Style / Modern C++ (10 problems)

41. Implement a **pipeline processing framework** using std::function and vectors.
42. Build **lazy evaluated vector transformations** (filter -> map -> reduce).
43. Implement a **template-based optional monad** with map, and_then operations.
44. Build a **compile-time string hashing function** using constexpr and templates.
45. Implement a **functional-style event dispatcher** supporting chained callbacks.
46. Build **lazy Cartesian product generator** for multiple containers.
47. Implement a **range-based sliding window generator** using iterators.
48. Build a **compile-time static lookup table generator** using templates.

49. Implement **constexpr regular expression matcher** for simple patterns.
50. Build a **pipeline-based JSON parser** using STL containers and functional transformations.
