### **Event: 100 Days CPP-Challenge**

### **Organizers: DevHive Club**

### **Department: CSE-Apex**

### **Institute: Chandigarh Engineering College (CEC), Jhanjeri**

### Week 1: **Basic Syntax and Structure**

**Objective**: Get familiar with C++ syntax, data types, and basic structures.

1. **Day 1**: Introduction to C++: Hello World, basic I/O (cin/cout), compiling a program.
2. **Day 2**: Variables and Data Types (int, float, char, string).
3. **Day 3**: Operators (arithmetic, comparison, logical).
4. **Day 4**: Conditionals (if, else, switch).
5. **Day 5**: Loops (for, while, do-while).
6. **Day 6**: Functions and Return Values.
7. **Day 7**: Recap & Practice problems (Basic calculator, number guessing game).

### Week 2: **Control Flow and Arrays**

**Objective**: Practice decision-making structures and work with arrays.

1. **Day 8**: Nested loops and control flow.
2. **Day 9**: Arrays: One-dimensional arrays.
3. **Day 10**: Multi-dimensional arrays (2D arrays).
4. **Day 11**: Array manipulation (sorting, searching).
5. **Day 12**: String manipulation and basic algorithms.
6. **Day 13**: Pass by reference vs. pass by value.
7. **Day 14**: Recap & Practice problems (Matrix multiplication, array rotation).

### Week 3: **Functions and Recursion**

**Objective**: Get comfortable with writing functions and recursion.

1. **Day 15**: Functions (declaration, definition, calling).
2. **Day 16**: Parameter passing (by value, by reference).
3. **Day 17**: Recursion: Basics and examples (factorial, Fibonacci).
4. **Day 18**: Tail recursion and optimization.
5. **Day 19**: Recursive problems (tower of Hanoi, reverse a string).
6. **Day 20**: Function overloading.
7. **Day 21**: Recap & Practice problems (recursive binary search, permutations).

### Week 4: **Object-Oriented Programming (OOP) Concepts**

**Objective**: Dive into object-oriented programming principles.

1. **Day 22**: Classes and objects (basic structure).
2. **Day 23**: Constructors and destructors.
3. **Day 24**: Access modifiers (private, public, protected).
4. **Day 25**: Member functions and data members.
5. **Day 26**: Static members and functions.
6. **Day 27**: Inheritance basics.
7. **Day 28**: Recap & Practice problems (Bank account class, Employee details).

### Week 5: **Inheritance and Polymorphism**

**Objective**: Advanced OOP concepts like inheritance and polymorphism.

1. **Day 29**: Inheritance: Single and multiple inheritance.
2. **Day 30**: Method overriding and virtual functions.
3. **Day 31**: Abstract classes and pure virtual functions.
4. **Day 32**: Operator overloading.
5. **Day 33**: Dynamic memory allocation (new, delete).
6. **Day 34**: Type casting and dynamic casting.
7. **Day 35**: Recap & Practice problems (Shape inheritance, polymorphic function calls).

### Week 6: **Templates and STL (Standard Template Library)**

**Objective**: Learn about templates and STL.

1. **Day 36**: Function templates.
2. **Day 37**: Class templates.
3. **Day 38**: Introduction to STL: Vectors.
4. **Day 39**: STL: Maps and Sets.
5. **Day 40**: STL: Algorithms (sort, find).
6. **Day 41**: STL: Iterators.
7. **Day 42**: Recap & Practice problems (Stack implementation using STL, map operations).

### Week 7: **File Handling and Exception Handling**

**Objective**: Understand how to manage files and handle exceptions.

1. **Day 43**: File handling: Reading and writing to files.
2. **Day 44**: File handling: Working with binary files.
3. **Day 45**: Exception handling basics (try, catch, throw).
4. **Day 46**: Custom exceptions.
5. **Day 47**: Working with file streams.
6. **Day 48**: Exception handling in functions.
7. **Day 49**: Recap & Practice problems (Reading/writing student data, exception handling in a calculator).

### Week 8: **Advanced Topics: Dynamic Memory and Smart Pointers**

**Objective**: Master memory management techniques.

1. **Day 50**: Pointers and references.
2. **Day 51**: Dynamic memory allocation.
3. **Day 52**: Smart pointers (unique\_ptr, shared\_ptr).
4. **Day 53**: Memory leaks and how to avoid them.
5. **Day 54**: Manual memory management vs. Smart pointers.
6. **Day 55**: RAII (Resource Acquisition Is Initialization) principle.
7. **Day 56**: Recap & Practice problems (Memory management in linked lists, resource cleanup).

### Week 9: **Data Structures: Linked Lists**

**Objective**: Get hands-on experience with linked lists.

1. **Day 57**: Introduction to linked lists.
2. **Day 58**: Singly linked list operations (insert, delete, traverse).
3. **Day 59**: Doubly linked lists.
4. **Day 60**: Circular linked lists.
5. **Day 61**: Linked list reversal and searching.
6. **Day 62**: Merging linked lists.
7. **Day 63**: Recap & Practice problems (Linked list palindrome, merge sorted lists).

### Week 10: **Advanced Data Structures: Trees**

**Objective**: Learn about trees and tree-based algorithms.

1. **Day 64**: Introduction to binary trees.
2. **Day 65**: Binary tree traversal (pre-order, in-order, post-order).
3. **Day 66**: Binary search trees (BST) operations.
4. **Day 67**: AVL Trees and balancing.
5. **Day 68**: Heaps (Min-heap, Max-heap).
6. **Day 69**: Trie (Prefix Tree) structure.
7. **Day 70**: Recap & Practice problems (BST insert/delete, tree traversal problems).

### Week 11: **Graphs and Algorithms**

**Objective**: Study graph theory and related algorithms.

1. **Day 71**: Introduction to graphs.
2. **Day 72**: Graph representations (adjacency matrix/list).
3. **Day 73**: Depth-first search (DFS).
4. **Day 74**: Breadth-first search (BFS).
5. **Day 75**: Shortest path algorithms (Dijkstra).
6. **Day 76**: Minimum spanning tree (Prim’s, Kruskal’s).
7. **Day 77**: Recap & Practice problems (graph traversal, shortest path).

### Week 12: **Advanced Algorithms and Techniques**

**Objective**: Focus on advanced algorithms and optimization techniques.

1. **Day 78**: Dynamic Programming: Basics and examples.
2. **Day 79**: Dynamic Programming: Fibonacci sequence, knapsack problem.
3. **Day 80**: Greedy Algorithms: Basics and examples.
4. **Day 81**: Divide and Conquer algorithms (Merge sort, Quick sort).
5. **Day 82**: Backtracking algorithms (N-Queens, Sudoku solver).
6. **Day 83**: Bit manipulation (XOR, Bit shifting).
7. **Day 84**: Recap & Practice problems (Knapsack problem, N-Queens).

### Week 13: **Final Projects and Challenges**

**Objective**: Apply your knowledge in real-world projects.

1. **Day 85-87**: Project 1: Build a text-based RPG (Use OOP, STL, file handling).
2. **Day 88-90**: Project 2: Create a personal finance manager (Implement dynamic memory, linked list, exception handling).
3. **Day 91-93**: Project 3: Design a simple database using C++ (File handling, exception handling).
4. **Day 94-96**: Project 4: Implement a Tic-Tac-Toe or Chess game with AI (Recursive algorithms, tree search).
5. **Day 97-99**: Final project refinement and optimization.
6. **Day 100**: Reflect on progress, improve code, and share your achievements.