**1. Fundamental Concepts**

* **Key Topics:**
  + Polymorphism (Compile time vs RunTime)
  + Overloading: function overloading vs operator overloading
  + Overriding: Function overriding, runtime polymorphism)
  + Function overriding, late binding, runtime type identification (RTTI)
  + Pass by value, Pass by address, Pass by reference
  + Pointers, references, arrays, temporaries, lvalue, rvalue
  + Deep copy, shallow copy
  + Access specifiers (public, private, protected)
  + Difference between class and struct
  + Inline vs macro vs constexpr
  + Difference between new and malloc, free and delete
  + Constructor sequence, destructor sequence
  + References and pointers: when to use which
  + Difference between static & global
  + Difference between encapsulation and abstraction
* **Sample Questions:**
  + What is the difference between deep copy and shallow copy?
  + Difference between memory leak and dangling pointer.
  + How does the destructor sequence work in inheritance?
  + Can you throw exceptions in constructors?
  + Can you make the constructor virtual?
  + When should you use pointers, const pointers vs references?

---

**2. Advanced Concepts**

* **Key Topics:**
  + Virtual function table, dynamic polymorphism
  + Managed vs unmanaged code
  + Memory leak detection in code
  + Reference counting in smart pointers or singleton classes
  + Placement new operator overloading
  + Restricting memory allocation on heap/stack
  + Multithreading
  + Interprocess communication (IPC)
* **Sample Questions:**
  + How does the virtual function table enable dynamic polymorphism?
  + What is the purpose of reference counting in smart pointers?
  + How can you detect memory leaks in C++?

---

**3. Interview Questions**

* **Key Topics:**
  + Difference between new and malloc, free and delete
  + Can a destructor be called explicitly?
  + How to overload operator **==**?
  + How to call a C function from a C++ program?
  + If a class is a "friend" to a base class, does it share the same "friend" property with derived classes?
  + In what circumstances can a copy constructor be called?
  + Can a reference be initialized to a constant value?
* **Sample Questions:**
  + What happens if a non-virtual function is overridden in a derived class?
  + How do you overload the **==** operator in C++?
  + What is the difference between public, private, and protected inheritance?

---

**6. C++ STL**

* **Key Topics:**
  + STL containers (vector, list, map, set, etc.)
  + STL algorithms (sort, find, transform, etc.)
  + Iterators and iterator categories
  + Allocators in STL
  + Functors and lambdas in STL
* **Sample Questions:**
  + What is the difference between std::vector and **std::list**?
  + How do you use std::sort with a custom comparator?
  + What are the advantages of using STL allocators?

---

**7. Modern C++**

* **Key Topics:**
  + C++11/14/17/20 features (e.g., **auto**, **constexpr**, **decltype**, **noexcept**)
  + Smart pointers (std::unique\_ptr, std::shared\_ptr, std::weak\_ptr)
  + Move semantics and rvalue references
  + Lambda expressions and closures
  + Concurrency and threading (**std::thread**, **std::async**)
* **Sample Questions:**
  + What is the difference between std::unique\_ptr and std::shared\_ptr?
  + How do move semantics improve performance in Modern C++?
  + What is the purpose of **std::async** in C++?

**8. Modern C++ Concurrency (New Section)**

* **Key Topics:**
  + **Concurrency Basics:**
    1. Threads (**std::thread**), thread lifecycle, and thread management.
  + **Synchronization Mechanisms:**
    1. Mutexes (std::mutex, **std::recursive\_mutex**), locks (std::lock\_guard, std::unique\_lock).
  + **Condition Variables:**
    1. std::condition\_variable for thread communication.
  + **Atomic Operations:**
    1. **std::atomic** for lock-free programming.
  + **Futures and Promises:**
    1. **std::future**, **std::promise**, and **std::async** for asynchronous programming.
  + **Thread Pools:**
    1. Implementing thread pools for efficient task management.
  + **Parallel Algorithms:**
    1. C++17 parallel STL algorithms (std::for\_each, std::reduce).
  + **Task-Based Concurrency:**
    1. Using **std::async** for task-based concurrency.
  + **Memory Model:**
    1. C++ memory model and std::memory\_order for atomic operations.
  + **Deadlocks and Race Conditions:**
    1. Identifying and resolving deadlocks and race conditions.
  + **Thread-Safe Data Structures:**
    1. Designing thread-safe containers and using concurrent data structures.
* **Sample Interview Questions:**
  + How do you create and manage threads in Modern C++?
  + What is the difference between std::lock\_guard and std::unique\_lock?
  + How does std::condition\_variable work, and when would you use it?
  + What is the purpose of **std::atomic**, and how does it ensure thread safety?
  + How do **std::future** and **std::promise** simplify asynchronous programming?
  + What are the advantages of using **std::async** over manually managing threads?
  + How do you implement a thread pool in Modern C++?
  + What are the key differences between sequential and parallel STL algorithms?
  + How do you prevent deadlocks in a multithreaded application?
  + What is the role of std::memory\_order in atomic operations?

---

**9. Design Patterns**

* **Key Topics:**
  + Creational patterns (Singleton, Factory, Builder)
  + Structural patterns (Adapter, Composite, Proxy)
  + Behavioral patterns (Observer, Strategy, Command)
  + Dependency injection in C++
  + Implementation of patterns using Modern C++ features
* **Sample Questions:**
  + How do you implement a thread-safe Singleton in C++?
  + What is the difference between the Factory and Builder patterns?
  + How can you use the Observer pattern in a multithreaded environment?

---

**10. System Design**

* **Key Topics:**
  + Designing scalable systems in C++
  + Memory management and optimization
  + Multithreading and synchronization (mutex, condition variables)
  + Interprocess communication (shared memory, message queues)
  + High-performance computing (cache optimization, SIMD)
* **Sample Questions:**
  + How do you design a thread-safe logging system in C++?
  + What are the best practices for memory management in a large-scale C++ application?
  + How do you implement producer-consumer using condition variables in C++?

**4. Qt-Specific Topics**

* **Key Topics:**
* **Types of Qt Projects:**
  + Widgets-based, Console-based, Dialog-based applications, SDI, MDI applications etc.
* **Qt Containers:**
  + Containers with reference counting and copy-on-write (e.g., QString, QByteArray, QVector).
* **Smart Pointers in Qt:**
  + QPointer, QSharedPointer, QWeakPointer.
* **Signals and Slots:**
  + Event-driven programming using signals and slots.
* **Qt .pro File:**
  + Project configuration (e.g., QT += core, QT -= gui, CONFIG += c++11).
* **Qt Widgets and Layouts:**
  + Widgets like QPushButton, QLabel, QLineEdit, and layouts like QVBoxLayout, QHBoxLayout.
* **Event Handling:**
  + Overriding event() and eventFilter() for custom event handling.
* **SDI and MDI Applications:**
  + Single Document Interface (SDI) vs. Multiple Document Interface (MDI).
* **Threading in Qt:**
  + QThread, QtConcurrent, and thread-safe signals/slots.
* **Qt Graphics Framework:**
  + QGraphicsView, QGraphicsScene, QGraphicsItem.
* **Qt Networking:**
  + QNetworkAccessManager, QWebSocket, QTcpSocket.
* **Qt Quick and QML:**
  + Declarative UI design using QML and integration with C++.
* **Sample Interview Questions:**
  + What is the difference between QPointer and QSharedPointer?
  + How does the signal-slot mechanism work internally in Qt?
  + What is the purpose of the .pro file in a Qt project?
  + How do you implement multithreading in Qt using QThread?
  + What is the difference between SDI and MDI applications in Qt?
  + How do you handle custom events in Qt using eventFilter()?
  + What are the advantages of using QML for UI design in Qt?
  + How does QNetworkAccessManager simplify HTTP requests in Qt?
  + **Sample Questions:**
  + What is the difference between SDI and MDI in Qt?
  + How does QSharedPointer implement reference counting?

**10. MFC (Microsoft Foundation Classes)**

* **Key Topics:**
  + MFC architecture and message handling
  + Document/View architecture in MFC
  + MFC controls (buttons, edit boxes, list controls, etc.)
  + Serialization in MFC
  + GDI and GDI+ in MFC
  + Event handling and message maps
  + Dialog-based applications in MFC
  + SDI and MDI applications in MFC
* **Sample Questions:**
  + What is the role of the message map in MFC?
  + How does the Document/View architecture work in MFC?
  + How do you handle custom events in MFC?
  + What is the difference between SDI and MDI applications in MFC?
  + How do you implement serialization in an MFC application

**5. Miscellaneous**

* **Key Topics:**
  + Build system (Make, CMake etc.)
  + UML diagrams, like Component Diagram, Sequence Flow Diagram, Class Diagram etc.
  + Architecture Diagrams.
* **Sample Questions:**
  + How do you find the middle element of a linked list with minimal time/space complexity?
  + What is the difference between clipping and culling in graphics?