National University of Computer and Emerging Sciences
Karachi Campus

Object Oriented Programming (CL-1004) Paper B

Lab Final-Exam

Total Time (hrs):	**02**
Total Marks:	**50**
Total Questions:	**03**

Date: **May 05, 2025**
Time: **08:20 AM – 10:20 AM**

Instructors
Mr. Talha Shahid, Mr. Nouman Hanif

Student Name	Roll No	Section	Student Signature

Do not write below this line

LLO 03: *Demonstrate advanced C++ OOP by implementing inheritance with access modifiers and virtual inheritance;* ***compile-time polymorphism*** *through function and* ***operator overloading; run-time polymorphism*** *via virtual functions, overriding,* ***abstract classes****, and* ***pure virtual functions****; and utilizing friend functions and classes for controlled flow.*

[wgt: 15 | 30 minutes]

Q1: You are part of a home-automation team at a consumer-electronics company that develops management software for a family of Samsung smart appliances. Your task is to design and implement the class hierarchy and operator overload so that the rest of the team can plug in new appliance types without worrying about the internals of power control or mode cycling.

1. **Classes to define**
   - `SamsungSmartAppliance`
     - **Data members:**
       - `modelName` (string)
       - `powerOn` (bool; starts OFF)
       - `currentMode` (int; index of current mode)
       - `numberOfModes` (int; total modes available)
     - **Member functions:**
       - **Constructor** - initializes all data members
       - `setPower(bool state)` — turns the appliance on or off (prints ON/OFF message)
       - `bool isPowerOn() const` — returns current power state
       - `displayStatus()` — prints model, power state, and (if ON) mode index
   - `SamsungSmartTV` (derived class)
     - **Data members:**
       - *(inherits all base members)*
     - **Member functions:**
       - **Constructor** (string name) — passes fixed TV mode count to base

- `displayStatus() const` — prints "Smart TV," model, power state, and current TV-mode name/index
- `SamsungWashingMachine` (derived class)
  - **Data members:**
    - *(inherits all base members)*
  - **Member functions:**
    - **Constructor** (string name) — passes fixed washer mode count to base
    - `displayStatus() const` — prints "Washing Machine," model, power state, and current wash-mode name/index

2. **Operator overload to provide**
   - **Prefix `operator++()`** in the base class.
     - **Purpose:** advance (and wrap) the appliance's mode only when it is powered on.

3. Deliverables

3.1 Class Definitions

- Declare and define all classes with their specified data members and member functions.

3.2 Operator Overload

- Implement the prefix `operator++()` overload in the base class (signature plus intended effect).

3.3 main() Demonstration

1. Create one `SamsungSmartTV` and one `SamsungWashingMachine`.
2. Call `displayStatus()` on both.
3. Toggle power with `setPower(true)`.
4. Use ++ to cycle modes and then call `displayStatus()` again.

LLO 03: *Demonstrate advanced C++ OOP by implementing inheritance with access modifiers and **virtual inheritance**; compile-time polymorphism through function and operator overloading; run-time polymorphism via virtual functions, overriding, abstract classes, and pure virtual functions; and utilizing **friend functions and classes** for controlled flow.*

[wgt: 15 | 30 minutes]

Q2: Suzuki Pakistan's Port Qasim plant produces cars, motorcycles, and hybrid vehicles (both fuel and electric). A `QualityControlSystem` requires direct access to private manufacturing metrics (e.g., engine power, torque) to ensure compliance with standards. Engineers use calibration tools to adjust vehicle parameters dynamically. Legacy code for hybrid vehicles inherits from both car and motorcycle classes, causing duplication in shared components like chassis IDs.

Your Task:

Design a C++ program to model this system. Each vehicle type must report its assembly status, and hybrid vehicles must combine metrics from both car and motorcycle components. The `QualityControlSystem` must directly inspect protected metrics, and calibration functions must adjust parameters without public setters. Design proper inheritance structure.

1. Class **`Vehicle`**
   - **Data Member:**
     - `modelID` (string, protected)
   - **Function:**
     - `void assembleVehicle()` – prints assembly initialization message.
2. Class **Car** (derived class)
   - **Data Member:**
     - `enginePower` (float, private)
   - **Function:**

- ○ `void assembleVehicle()` – method to print engine power.

3. Class **Motorcycle** (derived class)
   - **Data Member:**
     - ○ `torque` (float, private)
   - **Function:**
     - ○ `void assembleVehicle()` – prints `torque`.
4. Class **HybridVehicle**
   - **Function:**
     - ○ `void assembleVehicle()` –prints combined power using `enginePower` and `torque`.
5. Class `QualityControlSystem` (can **directly** access data members and member functions related classes)
   - **Function:**
     - ○ `void inspectCar(parameters)` – prints `modelID` and `enginePower`.
     - ○ `void inspectMotorcycle(parameters)` – prints `modelID` and `torque`.
6. **Non-Member Functions that directly access Protected Data**
   - `void calibrateEngine(parameters)` – modifies `enginePower` of `Car`.
   - `void calibrateTorque(parameters)` – modifies `torque` of `Motorcycle`.
7. In the `main()` function create proper objects, make function calls and test the system.

LLO 04: *Demonstrate proficiency in C++ generic programming and robust error management by implementing function and class templates, **handling exceptions,** and performing practical file operations and **IOStream-based input/output.***

[wgt: 20 | 60 minutes]

Q3: A city transit agency needs a lean C++ tool that logs daily ticket sales and revenue to disk, raises alerts whenever sales exceed capacity or revenue spikes past budgeted limits, and lets auditors quickly search back through past transaction records. Your task is to design classes—`DataFileHandler`, `TicketCountTransaction`, `TicketRevenueTransaction`, and `AlertSystem`—plus the exception types needed for robust error handling.

The `DataFileHandler` class holds a single filename member and offers three methods:

- `saveLine(paramters)` appends a CSV-formatted record (either ticket counts or revenue entries, or an alert message) and throws a `FileException` on write failure.
- `readAll()` prints every line in the file to `cout`, throwing `FileException` if the file can't be opened.
- `searchInFile(paramters)` scans for a keyword, prints matching lines (or reports none found), and throws a `SearchException` if the keyword is empty or the file read fails.

The `TicketCountTransaction` class encapsulates an int value (`tickets sold`) and a string `timestamp`, with `setTransaction(paramters)` rejecting negatives via `InvalidTicketException` and `getFormattedData()` returning "`timestamp,value`".
Similarly, `TicketRevenueTransaction` holds a `double` value (`revenue`) and a `string timestamp`, with the same two methods and the same exception on negative amounts.
The `AlertSystem` class keeps a reference to a `DataFileHandler` (for the alerts file) and exposes `checkThreshold(paramters)`, which logs "`[ALERT] Overbooked…`" `if ticketsSold > seatLimit and` "`[ALERT] Revenue spike…`" if `revenue > revenueLimit`.
Define an exception hierarchy with `of related classes`. Finally, **test your program** by instantiating the handler and transaction objects, calling `setTransaction()`, `saveLine()`, `checkThreshold()`, `readAll()`, and `searchInFile()` inside appropriate `try/catch` blocks so you can verify both normal and over-limit behavior as well as search functionality.

Q1 | Sample Output

```
--- Initial Status ---
Appliance: Smart TV, Model: QN90C Neo QLED, Power: OFF
Appliance: Washing Machine, Model: Bespoke WF53BB8, Power: OFF

--- Attempting Mode Change While Off ---
QN90C Neo QLED is powered off. Cannot change mode.

--- Powering On ---
QN90C Neo QLED powered ON.
Bespoke WF53BB8 powered ON.

--- Status After Power On ---
Appliance: Smart TV, Model: QN90C Neo QLED, Power: ON, Mode: Standard (1/3)
Appliance: Washing Machine, Model: Bespoke WF53BB8, Power: ON, Mode: Cotton (1/4)

--- Cycling Modes ---
QN90C Neo QLED mode changed.
Appliance: Smart TV, Model: QN90C Neo QLED, Power: ON, Mode: Movie (2/3)
Bespoke WF53BB8 mode changed.
Bespoke WF53BB8 mode changed.
Appliance: Washing Machine, Model: Bespoke WF53BB8, Power: ON, Mode: Delicates (3/4)
QN90C Neo QLED mode changed.
QN90C Neo QLED mode changed.
Appliance: Smart TV, Model: QN90C Neo QLED, Power: ON, Mode: Standard (1/3)
Bespoke WF53BB8 mode changed.
Bespoke WF53BB8 mode changed.
Appliance: Washing Machine, Model: Bespoke WF53BB8, Power: ON, Mode: Cotton (1/4)
```

Q2 | Sample Output

```
SWIFT-2025 Car: Engine power = 96 HP
[QC] Car SWIFT-2025: Engine = 96 HP
GS-150 Motorcycle: Torque = 10.5 Nm
[QC] Motorcycle GS-150: Torque = 10.5 Nm
HYB-1 Car: Engine power = 75 HP
HYB-1 Motorcycle: Torque = 15 Nm
HYB-1 Hybrid: Combined power = 90 units
```

Q3 | Sample Output

```
--- Contents of bookings.txt ---
2025-04-26T10:00:00,450
2025-04-26T10:00:00,9000.500000
2025-04-26T12:00:00,600
2025-04-26T12:05:00,12000.000000

--- Contents of alerts.txt ---
[ALERT] Overbooked: 600 tickets sold (limit 500)
[ALERT] Revenue spike: $12000.000000 (limit $10000.000000)

Enter keyword to search in bookings.txt: 9000

--- Search results for '9000' in bookings.txt ---
2025-04-26T10:00:00,9000.500000
```

bookings.txt	alerts.txt
2025-04-26T10:00:00,450 2025-04-26T10:00:00,9000.500000 2025-04-26T12:00:00,600 2025-04-26T12:05:00,12000.000000	[ALERT] Overbooked: 600 tickets sold (limit 500) [ALERT] Revenue spike: $12000.000000 (limit $10000.000000)