National University of Computer and Emerging Sciences
Karachi Campus

Object Oriented Programming (CL-1004)
Paper A

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: A coastal city uses tidal and geothermal power plants to generate electricity. Some plants combine both energy sources. A `MaintenanceSystem` needs direct access to protected operational metrics (e.g., output levels) of all plants for performance analysis. Engineers use calibration functions to tweak power output.

Your Task:
Design a C++ program to model this system. Each plant must report its energy generation, and hybrid plants must combine outputs from both sources. The `MaintenanceSystem` must directly inspect private metrics of all plants, and external calibration functions must adjust outputs without using public setters. Ensure hybrid facilities resolve conflicts in shared components.

1. Class `PowerSource`
   - **Data member:** `plantID` (string, protected).
   - **Function:** `void generateEnergy()` (simulates energy generation; outputs a message).
2. Class `TidalPlant`
   - **Data members:** `tidalOutput` (float, protected), `waveEfficiency` (float, protected).
   - **Function:** `void generateEnergy()` (prints tidal energy output and efficiency).
3. Class `GeothermalPlant`
   - **Data members:** `geoOutput` (float, protected), `steamEfficiency` (float, protected).
   - **Function:** `void generateEnergy()` (prints geothermal energy output and efficiency).
4. Class **HybridPlant**
   - **Function:** `void generateEnergy()` (calculates total output).

5. Class **MaintenanceSystem** (can **directly** access data members and member functions of related classes)
   - **Function: void `inspectTidal(----)`** (prints **`tidalOutput`** and **`waveEfficiency`**).
   - **Function: void `inspectGeothermal(----)`** (prints **`geoOutput`** and **`steamEfficiency`**).
6. **Non-Member Functions that directly access Protected Data:**
   - **void `adjustTidal(parameters)`** (modifies **`tidalOutput`**).
   - **void `adjustGeothermal(paramerters)`** (modifies **`geoOutput`**).
7. In the **`main()`** function create proper objects, make function calls and test the system.

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: You are part of a software team at a consumer-electronics company that develops management software for a family of Xiaomi devices. Your task is to design and implement the class hierarchy and operator overloads so that the rest of the team can plug in device types without worrying about the internals of energy consumption or output formatting.

1. **Classes to define**

   - **`XiaomiDevice`**
     - **Data members:**
       - **`modelName`** (string)
       - **`batteryLevel`** (int, 0–100) (starts with 100% battery)
     - **Member functions:**
       - **`performCoreFunction()`** (this function must be implemented by each derived class)
       - **`displayStatus()`** — prints model name and battery level
       - **`getBatteryLevel()`** — returns current battery level
   - **`XiaomiPhone`** (derived class)
     - **Data members:**
       - *(inherits* **`modelName`** *and* **`batteryLevel`***)*
     - **Member functions:**
       - Implements **`performCoreFunction()`** — simulates making a call and decrements battery by certain value and with proper validation and display current vatter level.
   - **`XiaomiSmartBand`** (derived class)
     - **Data members:**
       - *(inherits* **`modelName`** *and* **`batteryLevel`***)*
     - **Member functions:**
       - Implements **`performCoreFunction()`** — simulates activity tracking and reduces battery with proper validation and indication of current battery
2. **Operator Overloads to provide**
   - **`operator<<`**
     - Allows any **`XiaomiDevice`** (or subclass) display its status.
   - **`operator>`** between two **`XiaomiDevice`** references
     - Enables direct comparison of two devices based on their remaining battery level.

- Declare and define the above classes with the specified data members and member functions.
- Declare the two functions for `operator<<` and `operator>` in `XiaomiDevice`.
- In your `main()`, demonstrate:
  1. Creating one `XiaomiPhone` and one `XiaomiSmartBand`
  2. Calling `displayStatus()` via `operator<<`
  3. Calling `performCoreFunction()` on each
  4. Comparing them with `operator>` and printing which has more battery.

O 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 coastal research outpost needs a compact C++ tool that logs temperature and pressure readings to disk, raises alerts when those readings breach safety limits, and lets technicians quickly search past records for keywords. Your job is to design and implement classes—`DataFileHandler`, `TemperatureMeasurement`, `PressureMeasurement`, and `AlertSystem`—plus a `main()` function that ties them together.

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

- `saveLine(parameters)` appends a line (e.g. CSV data or an alert message) to the file, throwing a `FileException` on write failure.
- `readAll()` opens the file and prints every line to `cout`, throwing a `FileException` on read failure.
- `searchInFile(parameters)` scans the file for a given keyword, prints each matching line (or reports none found), and throws a `SearchException` if the keyword is empty or the file can't be opened.

The `TemperatureMeasurement` class stores two private members—a `double measurementValue` and a `string timestamp`—and provides:

- `setMeasurement(parameters)`, which rejects negative values via `InvalidMeasurementException` and sets both members, and
- `getFormattedData()`, which returns a `"timestamp,value"` CSV string.

Similarly, the `PressureMeasurement` class holds an `int measurementValue` and a string `timestamp`, with the same two methods (`setMeasurement(parameters)` throwing on negatives, and `getFormattedData()`).

The `AlertSystem` class keeps a reference to a `DataFileHandler` (for the alerts file) and exposes `checkThreshold(parameters)`, which compares `"TEMP"` readings against `50.0` or `"PRESSURE"` readings against `1100`, and writes an alert line (`"[ALERT] High TYPE detected: VALUE"`) via the handler when limits are exceeded.

Finally, write a simple `main()` that instantiates these classes, then test your program by creating the necessary objects and wrapping calls to `setMeasurement()`, `saveLine()`, `checkThreshold()`, `readAll()`, and `searchInFile()` in appropriate `try/catch` blocks to verify functionality and observe exception handling in action.

Q1 | Sample Output

```
COAST-1 Tidal: 91 MW (Eff: 65%)
[INSPECT] Tidal COAST-1: 91 MW, Eff 65%
VOLC-1 Geothermal: 100 MW (Eff: 75.5%)
[INSPECT] Geothermal VOLC-1: 100 MW, Eff 75.5%
HYB-1 Tidal: 90 MW (Eff: 68%)
HYB-1 Geothermal: 150 MW (Eff: 80%)
HYB-1 Hybrid Total: 240 MW
```

Q2 | Sample Output

```
--- Initial Device Status ---
Model: Xiaomi 14 Pro, Battery: 100%
Model: Mi Band 8, Battery: 100%

--- Using Devices ---
Making a call with Xiaomi 14 Pro...
  (Battery remaining: 90%)
Tracking activity with Mi Band 8...
  (Battery remaining: 95%)
Tracking activity with Mi Band 8...
  (Battery remaining: 90%)
Making a call with Xiaomi 14 Pro...
  (Battery remaining: 80%)

--- Current Device Status ---
Model: Xiaomi 14 Pro, Battery: 80%
Model: Mi Band 8, Battery: 90%

--- Comparing Battery Levels ---
Mi Band 8 has more battery than Xiaomi 14 Pro.

--- Demonstrating Polymorphism ---
Device 1 Status (via pointer): Model: Xiaomi 14 Pro, Battery: 80%
Device 2 Status (via pointer): Model: Mi Band 8, Battery: 90%
```

Q3 | Sample Output

```
--- Contents of scientific_data.txt ---
2023-10-27T10:00:00,25.500000
2023-10-27T10:05:00,1013
2023-10-27T10:10:00,55.000000
2023-10-27T10:15:00,1150

--- Contents of alerts.txt ---
[ALERT] High TEMP detected: 55.000000
[ALERT] High PRESSURE detected: 1150.000000

Enter keyword to search in scientific_data.txt: 25.50000

--- Search results for '25.50000' in scientific_data.txt ---
2023-10-27T10:00:00,25.500000
```

scientific_data.txt	alerts.txt
2023-10-27T10:00:00,25.500000 2023-10-27T10:05:00,1013 2023-10-27T10:10:00,55.000000 2023-10-27T10:15:00,1150	[ALERT] High TEMP detected: 55.000000 [ALERT] High PRESSURE detected: 1150.000000