NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES

CS-1004 — Object Oriented Programming (Spring 2025)

Semester Project: OSIM - Organizational Simulation and Internal Management System

Total Marks: 250 **Submission Deadline:** 9th May, 2025 at 11:59PM

Submission Guidelines

- Submit a `.zip` file named: `ROLLNUM1_ROLLNUM2_SECTION.zip`
  (e.g., 23i-1234_23i-5678_B.zip)
- Submission must include:
  - All `.cpp` and `.h` source files
  - CRD diagram in **PDF format**
- Do **NOT** include:
  - Executables (.exe)
  - System files (e.g., `.DS_Store`)
  - IDE-specific project folders
- Submit via Google Classroom by **9th May 2025, 11:59 PM**
  Late submissions will not be accepted
- All work must be original. Plagiarism will result in **zero marks** and disciplinary action

---

1. Project Overview

OSIM simulates an internal organizational environment where users with different roles (Junior, Manager, Director, Executive) interact through tasks, messages, and resources. All access is strictly governed by Role-Based Access Control (RBAC), clearance levels, and centralized policy enforcement. Key activities include task delegation, secure messaging, and audit logging, all with runtime accountability and secure operation tracking.

This project reinforces core Object-Oriented Programming (OOP) concepts including inheritance, polymorphism, file handling, memory management, and access control. The simulation also includes RBAC enforcement, audit logs, timeout monitoring, credential security, and role-sensitive communication.

---

2. Learning Objectives

- Implement multi-level inheritance and runtime polymorphism
- Enforce access control using role-based permissions
- Use only pointers and manual memory management (no STL)
- Overload operators to support task and resource handling
- Serialize/deserialize data to/from text and binary files
- Log all significant user behavior securely and immutably

---

3. Core Functional Requirements

A. User System & RBAC

- Class hierarchy: `Junior → Employee → Manager → Director → Executive` (Minimum 3 levels - l*eft means lowest*)
- Clearance levels mapped to each role and enforced globally
- Central `PolicyEngine` class must control all permission checks
- All access control must reference `PolicyEngine`; no hardcoded checks allowed

B. Authentication & Credential Security

- Users authenticate using credentials stored in `users.txt`
- Passwords must be ***stored as hashed*** using any custom method of your choice.
- Max 3 login attempts; access denied afterward

C. Task System

- Authorized users can create, assign, and delegate tasks
- Task statuses: `Created`, `Assigned`, `InProgress`, `Completed`, `Expired`
- Delegation allowed only between equal or higher clearance roles. (*means that a user can only delegate a task to someone who has the same or higher clearance level than themselves — never to someone with a lower clearance*)
- `TimeManager` must expire tasks after TTL (Time-To-Live)
- Tasks must record creator, assignee, timestamps, and status

D. Timeout & Inactivity Handling

- Each task must have a TTL (in seconds or minutes)
- TTL-based expiration must be runtime-evaluated using timestamps
- Recursive timeout checks to mark overdue tasks as `Expired`

E. Secure Messaging

- Message types: `INFO`, `PRIVATE`, `ALERT`
- `PRIVATE` messages must be encrypted (any method of your choice)
- Only intended recipient can decrypt and read `PRIVATE` messages
- Message routing must verify user clearance and message type
- Inboxes (`inbox.txt`) must store messages per user

Example Scenarios for Messaging:

- A **Junior** sends a **PRIVATE** message to a **Manager** → Allowed (encrypted, only Manager can see)
- A **Junior** sends an **ALERT** to a **Director** → Not allowed (Junior doesn't have clearance)
- A **Director** sends an **ALERT** to another **Director** → Allowed
- A **Manager** sends an **INFO** to all Juniors → Allowed

F. Audit Logging

- All key actions must be logged in `audit.txt`:
  - Login/logout
  - Task creation, assignment, status updates
  - Message transmission details
- Use `<<` operator to append logs in format: `[timestamp] USERNAME ACTION DETAILS STATUS`
- `audit.txt` must be append-only and immutable during execution

---

4. Additional Features

A. Performance Review System

- Tracks employee task completion, response to deadlines, and message interactions to generate a performance score.

- *Operator overloading (<<)* used to generate detailed performance reports.

B. Multi-Factor Authentication (MFA) Simulation

- After password login, a temporary OTP is generated and sent to the user via secure messaging.

- OTP must be entered within a certain time limit; otherwise, access is denied.

C. Audit Anomaly Detection

- Monitors audit logs for suspicious behavior (e.g., high task reassignment rates or unusual login times).

- Flags suspicious activities and triggers alerts, providing an anomaly report.

D. Global Notification System

- System-wide notifications can be sent to users, with types like WARNING and EMERGENCY.

- Only Managers and Executives can send certain types of notifications.

- Notifications are logged and stored in the system.

E. Task Priority System

- Tasks can be assigned priority levels: High, Medium, Low.

- Higher-priority tasks take precedence in delegation and management.

---

5. Optional Bonus Features

- Digital signatures for task approvals (hash of username + timestamp)
- Cycle detection in task delegation chains
- Encrypted binary log file.
- GUI-style dashboard using ASCII art and box formatting

---

6. Technical Constraints & Requirements

- Minimum **15 distinct classes** with clear OOP principles
- Must include:
    - Inheritance with at least 3 hierarchical levels
    - Runtime polymorphism using base class pointers
    - Operator overloading: `+=`, `<<`, `==`, `()`
    - Manual file handling: `users.txt`, `tasks.dat`, `audit.txt`, etc.
    - Recursion for TTL handling and delegation logic
    - No STL containers or smart pointers allowed