# Object-Oriented Programming (OOP) Sessional Project Plan

**Objective:**
Throughout the semester, students are required to develop an individual software project that progressively demonstrates the application of Object-Oriented Programming (OOP) concepts. Each laboratory session will emphasize a specific principle that must be appropriately integrated into the student's selected project scenario.

The laboratory manual includes:

- A **front-page** format for each report
- A **table of contents (index)** with lab numbers and teacher signature space
- **Detailed report writing instructions** (structure, format, and what to include)
- The content should be **typed or neatly handwritten**.

**N.B. (Important Notice)**

**All students must write every lab report, for example, and code** by their own understanding**. You are encouraged to take guidance from** class notes, lecture slides, and discussions with friends **to clarify concepts — but** not to copy any part directly**.**

Any report or program found to be **copied, AI-generated, or taken from seniors or online sources** will result in a **direct zero (0)** for that lab Report. No excuses, reconsideration, or partial marks will be given in such cases.

Students are expected to **learn, implement, and explain** their own Report. The primary goal of this course is **understanding**, not duplication.

## Each report must follow the following:

1. **Title and Objectives**

   - Clearly state the **Lab Report Title** (as given in the schedule).
   - Mention 2–3 **objectives** explaining what the student will learn or demonstrate in this lab.

2. **Theoretical Background**

   - Briefly explain the **concept or topic** covered (e.g., inheritance, abstraction).
   - Include **definitions**, **key properties**, or **use cases** relevant to the topic.
   - Diagrams or flowcharts are encouraged for visual understanding.

3. **Syntax and Explanation**

   - Write the **main syntax** related to the concept (e.g., syntax of class, constructor, function overloading).
   - Explain each part of the syntax in short, clear points.

4. **Source Code**

   - Present the **program code** neatly formatted and indented.
   - Include **comments** in the code to describe key sections.
   - Use consistent naming conventions and indentation.

5. **Screenshot of Execution**

   - Attach **clear screenshots** showing your program running successfully in the IDE or terminal.
   - Ensure your name, project file, or relevant output is visible.

6. **Observation and Result**

   - Summarize what you observed while executing the program.
   - State the **result** or conclusion derived from this lab.

# Mid Lab Topics

| Report No. | Lab Report Title | Instructions / Scenario |
|---|---|---|
| Lab Report 1 | Implementation of User-Defined Data Structures | Design and define essential data components of your project using **struct** to represent core entities (e.g., customer, item, sensor). |
| Lab Report 2 | Application of Union and Enumeration for Data Representation | Utilize **union and enum** to manage category-based data or variable data types within your project efficiently. |
| Lab Report 3 | Exploration of Storage Classes and Constant Qualifiers | Integrate **auto, static, and const** in your project code to ensure controlled memory usage and variable persistence. |
| Lab Report 4 | Exception Handling Mechanism in Project Modules | Introduce error-handling using **try, catch, and throw** blocks to manage runtime errors or invalid input in your project functions. |
| Lab Report 5 | File Handling for Persistent Data Management | Implement **file** operations for **storing, retrieving, and updating** data relevant to your project (e.g., records, logs, or configurations). |
| Lab Report 6 | Implementation of Data Mapping and CRUD Operations | Use **map** and related STL containers to **create, update, and delete** project-specific records dynamically. |
| Lab Report 7 | Function Overloading and Argument Handling in Modular Design | Apply **call-by-reference, function overloading**, and **default arguments** to develop reusable and flexible functions within your project. |
| Lab Report 8 | Design and Implementation of Classes and Encapsulation | Create **class structures to encapsulate** project attributes and methods, maintaining data security and abstraction. |
| Lab Report 9 | Advanced Class Features and Controlled Accessibility | Apply **this pointer, static data members, and friend functions** to optimize class-level behavior and access control. |

# Final Lab Topics

| Report No. | Lab Report Title | Instructions / Scenario |
|---|---|---|
| Lab Report 10 | Constructors, Copy Constructors, and Destructors in Object Lifecycle Management | Implement **constructors, copy constructors, and destructors** to handle object creation and cleanup processes within your project. |
| Lab Report 11 | Inheritance and Hierarchical Class Structures | Demonstrate different types of inheritance (**single, multiple, multilevel, hierarchical, and hybrid)** to promote reusability and modularity in your project. |
| Lab Report 12 | Method Overriding and Ambiguity Resolution Techniques | Customize **inherited class behaviors through method overriding** and apply **ambiguity resolution strategies** where required. |
| Lab Report 13 | Dynamic and Static Object Allocation | Create and manage project objects **dynamically (heap) and statically (stack)** to understand memory management and control. |
| Lab Report 14 | Abstraction and Interface Design using Virtual Functions | Design **abstract classes and interfaces** with virtual functions to enhance modularity and abstraction in your project architecture. |
| Lab Report 15 | Generalization and Specialization in Class Hierarchies | Apply class **generalization and specialization** principles to represent real-world relationships effectively in your project. |
| Lab Report 16 | Modeling Class Relationships: Association, Composition, and Aggregation | Implement inter-class relationships **(association, composition, and aggregation)** to reflect logical dependencies in your project system. |