Project Specifications (3% of Final Grade)

General Conditions for Project Development

- The application should run without runtime exceptions (avoid memory leaks and manage values properly with pointers).
- The project source code should compile without errors.
- Global variables or functions are not allowed.
- Each entity must have an associated class with logically connected methods and attributes.
- Classes must not be generic placeholders but must represent meaningful entities.
- Do not use `string`.

Gym Membership Management Application

The app will manage gym members, their subscriptions, attendance, and activity history. Key features include:

- Member registration, with options to add, edit, and delete member data.
- Membership plans (monthly, quarterly, annual) and associated payments.
- Attendance tracking with daily and monthly reports.
- Member activity tracking (workout types, duration, frequency).
- Historical data management using text or binary files.
- Generate reports such as:
  - Total revenue from memberships.
  - Attendance statistics by day/month.
  - Most popular workout types.
  - Member-specific activity summaries.

Phase 1: Initial Setup (1% Points)

1. Define at least **5 classes** related to the project:
   - **Member**: Details of a gym member (name, age, membership type, etc.).
   - **MembershipPlan**: Details of available membership plans (duration, price, benefits).
   - **Attendance**: Tracks daily attendance of members.
   - **Activity**: Tracks workout sessions (type, duration, calories burned).
   - **GymManager**: Central manager for handling members, plans, and activities.
2. Each class must include:
   - At least **two dynamically allocated fields** managed with a pointer.
   - At least **one static vector** (e.g., for storing attendance or activity logs).
   - Both **character arrays (char*)** and **numeric value vectors** (e.g., `int*`).
   - At least **one static field** and **one constant field**.
   - Private attributes with public accessor methods (getters/setters with validation).
   - Two generic methods for attribute processing.

- Two parameterized constructors, a copy constructor, destructor, and overloaded assignment (=).
- Overloaded operators:
  - Input (>>) and output (<<).
  - Indexing (`[]`), at least one arithmetic (+, -, etc.), increment/decrement (++, --), negation (`!`), conditional (<, >=, etc.), and equality (==).

Phase 2: Extra Features (1% Points)

1. Implement modules for:
   - **File input**: Accept text files via command-line arguments to populate data (e.g., `GymApp.exe members.txt plans.txt`).
   - **Binary file storage**: Save all session data (members, attendance, activities). If no input files are given, load from default binary files.
   - **Reports**: Provide at least 3 reports such as attendance by date, revenue, and activity trends.
   - **Menu system**: Navigate the application using numeric or text-based options (e.g., add a new member, view attendance, generate reports).
2. Implement an example of **class composition** (e.g., `GymManager` contains objects of `Member` and/or `MembershipPlan`).
3. Extend classes through **inheritance**, without modifying existing classes (e.g., `PremiumMember` inherits from `Member` and adds new characteristics/behaviors).
4. Create an **abstract class** (e.g., `ReportGenerator` with virtual methods for generating reports).
5. Implement at least **2 non-pure virtual methods** in other classes.

Phase 3: Advanced OOP Concepts (1% Points)

1. Implement a **template class** named `GymStorage<T>` that acts as a container for storing data of any type (`T`).
2. Use an **STL vector of pointers** to manage elements of a class hierarchy (e.g., a vector of `Member` pointers containing both `RegularMember` and `PremiumMember` instances).

Observations:

- **Testing of classes** is mandatory to receive the full points.
- **Submission Format**: Submit the project as a single `.cpp` file following the naming conventions provided:
  `[LastName]_[FirstName]_G[GroupId].cpp`
- **Anti-Plagiarism Verification**: The code will be checked for originality. Copying more than 30% will result in penalties as outlined in the guidelines.
- **Presentation**: Each student will present their project (3–5 minutes) in an online session during the last week of the semester. We will agree together on the presentation date during the first seminar after vacation. Students who do not present their project will receive **0 points**.