Coding Assignment 1: C++ Review
Due: Wed Feb 11, 2026 11:59pmDue: Wed Feb 11, 2026 11:59pm
Ungraded, 100 Possible Points100 Points Possible
Attempt
In Progress
NEXT UP: Submit Assignment
Unlimited Attempts Allowed
Available: Jan 28, 2026 12:00am until Feb 18, 2026 11:59pmAvailable: Jan 28, 2026 12:00am until Feb 18, 2026 11:59pm

Coding Topic: VehicleManageSys

Objective

Students will learn to implement **object-oriented programming (OOP) concepts** in C++ by creating a simple **Vehicle Management System**.
This activity will reinforce the following concepts:

- Class creation and encapsulation
- Constructors, getters, and setters
- Typedef and enumerations
- Dynamic memory allocation (`new` and `delete`)
- Array of pointers to objects
- Sorting with custom criteria

Prerequisites

Before attempting this activity, students should understand:

- C++ class and object fundamentals
- How to define and use enumerations
- Dynamic memory allocation with `new` and `delete`
- Basics of sorting (`std::sort` or manual sorting algorithm)

Step-by-Step Instructions

Step 1: Use the Starter Code

Download and review the provided **C++ starter code** available at this link:
👉 [C++ Starter Code Link Here Download C++ Starter Code Link Here]

This starter code contains the basic structure of the program, including:

- `Vehicle` class definition
- Typedef for VIN
- Enumerations for Color and PurchaseMonth
- Example of dynamic allocation and sorting setup

Also, use can use the Codes from previous codes from .zip file link Download link

---

Step 2: Review the Code Structure

Open **VehicleManageSys.cpp** in your preferred editor.
You should see:

- **Vehicle class** with private attributes and public methods.
- **Typedef**: `VIN` defined as a fixed 20-character array.
- **Enumerations**: `Color` and `PurchaseMonth`.
- **Dynamic Array of Pointers**: Array of `Vehicle*`.
- **Sorting Functions**: Sorting by maker, color, and year.

Step 3: Understand the Vehicle Class

The `Vehicle` class includes:

- **Attributes**: `maker`, `model`, `year`, `color`, `weight`, `VIN`, `purchaseMonth`.
- **Constructor**: Initializes attributes.
- **Getters/Setters**: For each attribute.
- **Display Method**: Prints details of a vehicle.
- **Helper Methods**: Convert enums (Color, PurchaseMonth) to strings.

Step 4: Modify the Program (30 points)

1. **Add a New Attribute:**
    - Add `engineSize` (in liters, e.g., 2.0) to the `Vehicle` class.
    - Update constructor, getters, setters, and `display()` method.
2. **Add a New Sorting Criterion:**

    - Implement sorting by `weight`.
    - Add a comparison function and allow the user to choose sorting criteria in `main()`.

Step 5: Add New Vehicles (20 points)

- Extend the program so the user can **add new vehicles** at runtime.
- Prompt the user for: maker, model, year, color, weight, VIN, purchase month, and engine size.
- Dynamically allocate `Vehicle` objects using `new` and add them to the array.

Step 6: Compile and Run

1. Compile and do debugging
2. Run the program
3. Verify that:
    - Vehicles are displayed.
    - Sorting works by maker, color, year, and weight.
    - User can add new vehicles dynamically.
4. Take screenshot on results

Step 7: Submission (50 points)

Submit the following:

1. Modified **source code (.cpp)** file with comments.
2. Screenshot(s) showing successful program execution and sorted outputs.

Grading Rubric

- **30 pts –** Modification (new attribute + sorting criterion)
- **20 pts –** Adding new vehicles dynamically
- **50 pts –** Code quality, style, comments, and correct execution (with screenshots)