ADP C++ Object-Oriented Programming Assignment # 1

Course: ADP Computer Science
Subject: Object-Oriented Programming
Topic: C++ Classes and Real-World Problem Solving

Assignment Instructions

This assignment is divided into two parts.

- **Part 1 is a practical programming task.** You are required to design and implement a C++ application based on the problem description below. This part must be submitted as a single C++ source file (`.cpp`).
- **Part 2 is a theoretical, handwritten task.** You must answer the questions on paper. Your answers should be based on the program you created in Part 1. This part will not be solvable without completing the practical implementation first.

Part 1: Practical Task - Employee Management System

Problem Scenario:

A small software company wants to digitize its employee records. They need a simple console-based C++ application to manage basic employee information. As a programmer, your task is to develop this system using Object-Oriented Programming principles. The system should allow for adding new employee records and displaying the details of all employees.

Detailed Requirements:

You must implement a C++ program that fulfills the following specifications.

1. **`Department` Enumeration:**
   - Create an `enum` named `Department` with the following values: `HR`, `Engineering`, `Marketing`, `Finance`.
2. **`JoiningDate` Structure:**
   - Create a `struct` named `JoiningDate` to store the date an employee joined the company.
   - It should have three integer members: `day`, `month`, and `year`.
3. **`Employee` Class:**
   - Create a class named `Employee`.
   - **Private Members:**
     - `employeeID` (integer): A unique ID for each employee.

        - `employeeName` (string): The full name of the employee.
        - `dateOfJoining` (`JoiningDate` struct): The date the employee joined.
        - `department` (`Department` enum): The department the employee works in.
    - **Public Members:**
        - **A default constructor:** Initializes `employeeID` to 0 and `employeeName` to an empty string.
        - **`inputEmployeeData()`:** A member function that prompts the user to enter all details for an employee (ID, Name, Joining Date, and Department) and sets the corresponding member variables. This function will handle user input.
        - **`displayEmployeeData()`:** A member function that prints all the details of an employee to the console in a clean, readable format. This will handle the output.
        - **`getDepartment()`:** A member function that returns the `Department` of the employee.
        - **`getEmployeeID()`:** A member function that returns the `employeeID`.
4. **`displayByDepartment` Function (Function with Object Argument):**
    - Create a standalone function (not a member of the `Employee` class) named `displayByDepartment`.
    - This function should take two arguments: an array of `Employee` objects and the total number of employees.
    - Inside this function, prompt the user to select a department and then display the records of only those employees who belong to the chosen department.
5. **`main()` Function:**
    - The `main` function should be the driver of your program.
    - Declare an array of `Employee` objects (e.g., of size 50).
    - Implement a menu-driven interface that allows the user to:
        1. Add a new employee record.
        2. Display all employee records.
        3. Display employees by department.
        4. Exit the program.
    - Use a loop to keep the program running until the user chooses to exit.

Part 2: Theoretical Questions (To be submitted in written format)

Instructions: After completing your C++ program in Part 1, answer the following questions on paper. Your answers must specifically refer to the code you have written.

Question 1: Justification of Design Choices
In your `Employee` class, you have used both a `struct` (`JoiningDate`) and an `enum` (`Department`).

- a) Explain the primary difference between a `class` and a `struct` in C++.
- b) Justify your decision to use a `struct` for `JoiningDate` instead of a `class`.
- c) What was the main advantage of using an `enum` for the `Department`?

Question 2: Encapsulation and Data Hiding
The principle of encapsulation is fundamental to OOP.

- a) Define encapsulation in your own words.
- b) How have you implemented encapsulation in your `Employee` class?
- c) Why is it a good practice to keep the data members (`employeeID`, `employeeName`, etc.) `private`?

Question 3: Analysis of Member Functions
Analyze the member functions you created for the `Employee` class.

- a) Describe the purpose of the `inputEmployeeData()` member function. How does it interact with the `private` data members of the class?
- b) Explain the role of the `getDepartment()` member function. Why is it necessary to have such a function to access the `department` from outside the class?

Question 4: Functions and Objects
You created a standalone `displayByDepartment` function that takes an array of `Employee` objects.

- a) Explain how an `Employee` object is passed to this function in your implementation (e.g., by value or by reference).
- b) What are the advantages or disadvantages of the method you chose for passing the objects?

Question 5: Real-World Extension
Based on your implementation in Part 1, propose and describe **one significant improvement** to your `Employee` class.

- a) What new data member(s) and/or member function(s) would you add?
- b) Explain how this addition would make your Employee Management System more useful in a real-world scenario.