MTH 4300: Algorithms, Computers, and Programming II

HW #3

Due Date: October 30th, 2025

Problem 1

You need to create a class called `Book` that represents a book in a library. The class should include the following requirements and make use of the C++ concepts listed below:

Data Members

- **Private data members:**
  - `title` of type `std::string`
  - `author` of type `std::string`
  - `yearPublished` of type `int`
  - `price` of type `double`

Constructor

The class should have a constructor that takes the following parameters:

- `bookTitle` (a `std::string` passed by reference) for the title of the book
- `bookAuthor` (a `std::string` passed by reference) for the author of the book
- `publishedYear` (an `int` with a default value of 1900) for the year the book was published
- `bookPrice` (a `double` with a default value of 0.0) for the price of the book

Use an initialization list to initialize all the data members.

Methods

- Implement a method called `applyDiscount()` that takes a `double` discount percentage by reference and applies it to the price of the book.
- Implement a method called `getBookInfo()` that returns the book's details (title, author, year published, and price) as a formatted string. This method should be marked as `const` since it does not modify the object's state.

Example Usage

```
string bookName = "The Great Gatsby";
string author = "F. Scott Fitzgerald";
Book myBook(bookName, author, 1925, 15.99);

double discount = 10.0; // 10% discount
myBook.applyDiscount(discount);
myBook.getBookInfo();
```

Implementation Steps

- Define the `Book` class with the required private data members.
- Implement the constructor using an initialization list with default arguments.
- Implement the `applyDiscount()` method using pass-by-reference for the discount parameter.
- Implement the `getBookInfo()` method, ensuring it is marked as a `const` member function.

Your Task

Write the full implementation of the `Book` class according to the above specifications.

Problem 2

Create a class for a 3 by 3 matrix (using arrays and not vectors) named `Matrix33`:

- Make sure the private attribute is a 2D array: `double matrix[3][3];`
- A constructor that accepts a 2D array as an input parameter
- Add a default constructor that takes no arguments and does nothing in the body:

  ```
  Matrix33() {}
  ```

- Overload `*` operator for matrix multiplication
- Overload `*` operator for scalar multiplication
- Overload `+` operator for matrix addition
- Overload `<<` operator to print matrix
- Overload `>>` operator and prompt user to enter 9 consecutive values
- Write a function to compute the determinant of the matrix
- Make sure to separate the interface and implementation

Problem 3

- Modify the file `3d_point.cpp(algorithms_and_programming_mth4300/class_notes/7_operator_overload/pract` file we went over in class to separate the interface and implementation and rename it `Vector3`.
- Create a separate `main.cpp` file where you include the headers for `Matrix33` and `Vector3`.
- Overload the operator `()` for accessing the private attributes of the `Vector3` and `Matrix33` classes:

```
double operator()(int row, int col)
{
    return matrix[row][col];
}
```

- Write a function in `main.cpp` that takes a `Matrix33 = A` and `Vector3 = x` as input parameters and computes $Ax = b$, returning a `Vector3(b)`.
- Prompt the user to enter a 3x3 matrix and a 3D vector, then call your function to compute the product and print the result.

Problem 4

Do Problem 2, but for an $n \times m$ matrix using the vector template class (STL). In the constructor, add parameters for the number of rows (n) and number of columns (m).