Assignment 1

C++ Advanced Programming Assignment: Library Management System

Objective:

Develop a Library Management System using C++ that utilizes advanced concepts including Templates & STL, Class Templates, Function Templates, Inheritance, Polymorphism, and other core features. The system will manage a collection of books, track borrowing activities, and perform various operations on the library's catalog using STL containers and algorithms.

Requirements:

1. **Templates & STL**
    - Use Function Templates to create a search function that can search for a book by different attributes (title, author, ISBN).
    - Use Class Templates to define a generic class Catalog that manages a collection of items (books).
2. **Container Types**
    - Use STL containers (std::vector, std::list, std::map, etc.) to store and manage books and borrowing records.
3. **Iterators**
    - Use iterators to traverse through the container elements.
4. **Algorithms**
    - Use STL algorithms (e.g., std::sort, std::find) to perform operations on the library data.
5. **Functors**
    - Implement functors for custom operations such as sorting and filtering books.
6. **Inheritance & Polymorphism**
    - Create a base class LibraryItem and derive a Book class from it.
    - Implement polymorphic behavior for displaying item details using virtual functions.

7. **Abstract Class vs. Concrete Class**

    - Define an abstract class LibraryItem with pure virtual functions.
    - Implement concrete classes Book and Magazine.
8. **Constructors, Destructors, and Virtual Functions**
    - Implement constructors, copy constructors, destructors, and virtual destructors where necessary.
    - Show polymorphism by converting derived pointers to base pointers.

Assignment Description:

1. **Class Definitions:**
    -
    - Define a base class LibraryItem with common attributes like title, author, and ISBN. Include pure virtual functions like displayDetails().
    - Create a derived class Book that adds additional attributes like genre and year.
    - Create another derived class Magazine for additional practice.

2. **Template Classes:**

    - Implement a class template Catalog<T> to manage a collection of LibraryItem objects. The class should provide methods to add, remove, and search items.
3. **Function Templates:**

    - Implement a function template searchItem that can search for items by different attributes using function overloading.
4. **STL Containers and Algorithms:**

    - Use std::vector to store a list of Book objects in Catalog.
    - Use std::map to track borrowed books with keys as book IDs and values as borrower details.
    - Use STL algorithms like std::sort to sort books by title or year.
5. **Functors:**

    - Implement functors for sorting books by different criteria (e.g., by title, by author).
6. **Polymorphism:**

- Implement a virtual displayDetails function in the LibraryItem class and override it in Book and Magazine classes.
- Show polymorphic behavior by storing pointers to LibraryItem in the Catalog.

7. **Constructors and Destructors:**

- Implement necessary constructors and destructors, including virtual destructors in the base class.
- Implement a copy constructor for Catalog.

8. **Main Function:**

- In the main function, create instances of Catalog<Book> and Catalog<Magazine>.
- Add, search, and display items using the created instances.
- Demonstrate the use of function templates, class templates, iterators, algorithms, and functors

```
class Book : public LibraryItem {
private:
    std::string title;
    std::string author;
    std::string ISBN;
    std::string genre;
    int year;
public:
    Book(const std::string& title, const std::string& author,
const std::string& ISBN, const std::string& genre, int year);
    void displayDetails() const override;
    // Getters for attributes to be used in functors and
templates
    std::string getTitle() const;
    std::string getAuthor() const;
    std::string getISBN() const;
    int getYear() const;
    // Implement constructors and other necessary member
functions
};
```

```
// Template class for Catalog
template <typename T>
class Catalog {
private:
    std::vector<T> items;
public:
    void addItem(const T& item);
    void removeItem(const T& item);
    template <typename Func>
    T searchItem(Func func);
    void displayAll() const;
    // Implement other necessary member functions
};

// Functor for sorting by title
struct SortByTitle {
    bool operator()(const Book& a, const Book& b) {
        return a.getTitle() < b.getTitle();
    }
};

// Implement function templates for searching
template <typename T>
T searchItem(const std::vector<T>& items, const std::string&
key);

template <typename T>
T searchItem(const std::vector<T>& items, int key);

// Main function
```

```
int main() {

    Catalog<Book> bookCatalog;

    Book b1("The Great Gatsby", "F. Scott Fitzgerald",
"123456789", "Fiction", 1925);

    Book b2("1984", "George Orwell", "987654321", "Dystopian",
1949);


    bookCatalog.addItem(b1);

    bookCatalog.addItem(b2);


    // Sort books by title

    std::sort(bookCatalog.begin(), bookCatalog.end(),
SortByTitle());


    // Display all books

    bookCatalog.displayAll();


    // Search for a book by title

    Book foundBook = searchItem(bookCatalog,
std::string("1984"));

    foundBook.displayDetails();


    return 0;

}
```

This skeleton is a starting point. You will need to fill in the member function implementations, ensure proper memory management, and expand the functionality as required. Ensure that you thoroughly test your code with various scenarios to validate its correctness and robustness..

Part 2: Understanding Concepts

5. **Conceptual Questions:**

Answer the following questions based on the concepts covered in this assignment:

1. Explain the role of virtual functions and polymorphism in the Library Management System. Provide a detailed description of how virtual functions enable polymorphism and give an example from the provided code.

2. Additionally, discuss the importance of having a virtual destructor in the base class. What would happen if the destructor was not declared as virtual?
3. Create a UML class diagram.

Submission Guidelines:

Do not submit a zip folder

- Submit the C++ source code files (.cpp/.h) that includes all parts of the assignment.
- Submit a word document including the following content.
    1. Complete code
    2. Answers to any logical/conceptual questions.
    3. Output Screenshots
- Include a README file that briefly explains the program and how to compile and run it.

Evaluation Criteria:

- Correctness of the program.
- Proper use of C++ syntax and features.
- Code readability and comments.
- Implementation of the specified features and functionalities.

By completing this assignment, you will demonstrate your ability to integrate various C++ programming concepts into a single, cohesive project. Good luck!