Assignment 1

=======================

**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.**

**Answer: Role of Virtual Functions and Polymorphism:**

Virtual functions are essential for enabling polymorphism in C++. They allow subclasses to implement their own versions of functions declared in the base class. When a virtual function is invoked through a base class pointer, the actual method executed is determined at runtime based on the specific object referenced, not the type of the pointer itself.

In the Library Management System, the LibraryItem class includes a virtual method called displayDetails(). This method is overridden in the Book and Magazine classes to display details pertinent to each type. Consequently, if a LibraryItem\* pointer references a Book, calling displayDetails() will execute the version defined in the Book class.

Polymorphism allows the system to manage various item types, such as books and magazines, consistently while enabling each type to operate according to its specifications. For instance, when the catalog displays all items using catalog.displayAll(), it invokes displayDetails() for each item. Despite being stored as LibraryItem, the corresponding version of the function, whether from Book or Magazineis automatically executed.

This design enhances flexibility and simplifies adding new item types without changing the catalog functionality. Overall, polymorphism contributes to a clean and maintainable codebase by facilitating the introduction of additional item types with minimal impact on existing 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?**

**Answer: Importance of Virtual Destructor:**

A virtual destructor in a base class is crucial because it ensures that when a derived class object is deleted via a base class pointer, the derived class's destructor is called. Without a virtual destructor, only the base class's destructor would execute, which can lead to resource leaks.

If the LibraryItem class lacks a virtual destructor, deleting an instance of Book through a LibraryItem\* pointer results in undefined behavior, as only the LibraryItem destructor will run, leaving resources allocated in the Book class unreleased.

The presence of a virtual destructor guarantees that when a derived class object (such as Book or Magazine) is deleted through a base class pointer, the appropriate destructor for the derived class is invoked. This prevents improper cleanup of derived class resources.

For example, if a LibraryItem\* points to a Book and is deleted without a virtual destructor, the cleanup for the Book parts won't occur. This can lead to memory leaks, as only the base class's destructor performs cleanup, neglecting necessary cleanup for the derived class. Implementing a virtual destructor ensures that both the base and derived class destructors are executed, fully deallocating memory and preventing memory-related issues.

**3. Create a UML class diagram.**

**Answer:**

**A diagram of a computer

Description automatically generated**

**4. Screenshots of the output:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated