Lab 9

**Lab Exercise**

In this lab, you’ll create a Document Editor which supports components text, image, and table elements. You will implement a factory pattern to create both "new" and "old" versions of these components, utilizing the Abstract Factory and Composite patterns. You are to implement:

### Header Files

* **ComponentFactory.h**:
  + Define abstract and concrete factories for creating document components.
  + Implement two concrete factories:
    - OldComponentFactory (produces "old" versions of components)
    - NewComponentFactory (produces "new" versions of components)
* **DocumentComponent.h**:
  + Define abstract and concrete classes for document components.
  + Implement the following classes:
    - Text - Represents a text component.
    - Image - Represents an image component.
    - Table - Represents a table component.
    - CompositeDocument - Acts as a container for multiple components, allowing you to treat a group of components as a single unit.

### CPP Files

* **NewComponentFactory.cpp** and **OldComponentFactory.cpp**:
  + Implement the createText(), createImage(), and createTable() methods in each factory to produce new or old versions of each component type.

### Additional Notes

* **DocumentComponent**: Design an abstract DocumentComponent class and create specific classes for each component type (Text, Image, Table). The CompositeDocument class should also implement DocumentComponent, allowing it to contain other document components.
* **ComponentFactory**: Define an abstract factory interface for creating components (createText(), createImage(), createTable()). Implement concrete factory classes (OldComponentFactory and NewComponentFactory) that produce the "old" and "new" versions of the components.

**Submission Guidelines**

* C++ Code
* screenshot of output