Lab 3

You will use the following code to create 3 classes.   
The base class has been partially created for you (implementation of the member functions are required.

You will be using the base class “Plant” to create two inherited (derived) classes: Flower and Tree.

**Instructions**

**Completing the Plant Base Class:**

1. **Default Constructor (Plant::Plant()):**
   * Implement the default constructor to initialize the **Name** attribute. You can set it to **nullptr** as shown in the previous code samples.
   * Ensure that you correctly allocate and deallocate memory for the **Name** attribute in both the default constructor and the parameterized constructor.
2. **Copy Constructor (Plant::Plant(const Plant& other)):**
   * Implement a copy constructor that creates a deep copy of the **Name** attribute. This involves allocating new memory and copying the contents of the other object's **Name** attribute.
3. **Copy Assignment Operator (Plant& operator=(const Plant& other)):**
   * Implement the copy assignment operator to allow proper assignment of one **Plant** object to another. Ensure it makes a deep copy of the **Name** attribute, just like the copy constructor.

**Building the Flower Class:**

1. **Constructor (Flower::Flower(const string& name, const string& flowerColor)):**
   * Implement the constructor for the **Flower** class that initializes the **Name** attribute (inherited from the base class) and the **color** attribute specific to the flower.
2. **Display Function (Flower::display(ostream& os) const):**
   * Implement the **display** function to display information about the flower, such as its name and color.
   * You can access the **Name** attribute using the **getName** method inherited from the **Plant** class and the **color** attribute specific to the **Flower** class.

**Building the Tree Class:**

1. **Constructor (Tree::Tree(const string& name, int treeAge)):**
   * Implement the constructor for the **Tree** class that initializes the **Name** attribute (inherited from the base class) and the **age** attribute specific to the tree.
2. **Display Function (Tree::display(ostream& os) const):**
   * Implement the **display** function to display information about the tree, such as its name and age.
   * You can access the **Name** attribute using the **getName** method inherited from the **Plant** class and the **age** attribute specific to the **Tree** class.

You can use the Plant.h file to start with the base class.

Below is the UML for the derived classes.

|  |  |
| --- | --- |
| +-----------------+  | Flower |  +-----------------+  | - color: string |  +-----------------+  | + Flower(name: string, flowerColor: string) |  | + display(ostream& os) const |  | + getName() const |  | + setName(name: string) |  +-----------------+  | Inherited from |  | Plant |  +-----------------+ | +---------------+  | Tree |  +---------------+  | - age: int |  +---------------+  | + Tree(name: string, treeAge: int) |  | + display(ostream& os) const |  | + getName() const |  | + setName(name: string) |  +---------------+  | Inherited from |  | Plant |  +---------------+ |

Get and set name functions are inherited. No need to re-make them.