1. **In C++, how can I inherit a class's implementation without inheriting its interface? See the video lecture on Multiple Inheritance if you are having difficulty or see my online notes.**
    - When creating a subclass, you use protected instead of public for the base class
2. **Provide 2 reasons why object-oriented programs run more slowly than procedural programs. See the video lecture on Inheritance Implementation if you are having difficulty.**
    - It has small method bodies which could cause the doubling of the overhead to be significant
    - Virtual methods prevent a compiler from inlining methods
3. **Answer each of the following questions about modules:**

    a) **Name two benefits of modules (hint: if you are having trouble, then check part b below--it might help).**

    - You can access private and protected instance variables of a class
    - All variable, function, and class names end up in the same global name space

    b) **C++ uses two features to implement its module mechanism. Name those two features, and match them to the two benefits you described in part (a) (i.e., for each one C++ feature, indicate which benefit it provides). Check out the video lecture on Module Mechanisms in C/C++ or check my online notes if you are having difficulties.**

    - Friends – Access private and protected instance variables
    - Namespaces – Allows variable, function, and class names to exist in the same global name space

4. **Answer the following questions:**

    a) **Declare a variable called myListener to be of type ActionListener and initialize it with an instance of Actor.**

    - ActionListener myListener = new Actor();

    b) **Can myListener call actionPerformed? Why or why not?**
    - Yes, it is implemented in the actor class

    c) **Can myListener call displayBanner? Why or why not?**
    - No, the ActionListener variable type cannot see the displayBanner function

5. **Answer the following questions:**

   a) **What is the problem with the class hierarchy?**

   - Subclasses do not want to inherit certain methods from its superclass

   b) **Suggest another way to implement the oval and circle subclasses that allows you to "inherit" the implementation of the arc subclass without incurring the disadvantages of actual inheritance.**
   - Using composition – including arc as an instance variable to inherit its behavior

   Sketch out a sample Java class declaration for a circle that includes the following elements:

   i. **declarations for the instance variable(s)**
   ii. **declarations for the public methods (it is okay to use my "some parameters" notation for the parameters to the Draw method)**
   iii. **implementations for the setLeft and setDiameter methods.**

```
Class Circle {
    Oval data = new Oval();
    Public void Draw(some parameters) {
        data.Draw(some parameters);
    }
    Public void setLeft(int left) {
        data.setLeft(left);
    }
    Public void setTop(int top) {
        data.setTop(top);
    }
    Public void setDiameter(int diameter) {
        data.setWidth(diameter);
        data.setHeight(diameter);
    }
}
```

6. Answer the following yes/no questions about the above code and for each answer explain why you answered as you did:

    a. Is it legal to access the `value` variable in statement 1?
        - Yes. You can access protected members if they are in the same package.
    b. Is it legal to access the `name` variable in statement 2?
        - Yes. The default is package level protection, and they are in the same package.
    c. Is it legal to access the `header` variable in statement 3?
        - No. Queue is in a different package and therefore cannot access the package-protected variable.
    d. Is it legal to access the `sentinelNode` variable in statement 4?
        - Yes. Queue can access the protected variables of List since it is a subclass.
    e. Is it legal to access the `value` variable in statements 3&4 (even if you answered no to either statement 3 or 4, assume that you had answered yes and consider whether based on a "yes" answer, if `value` would be accessable)? **Hint the answer and reason is the same in both cases.**
        - No, it is not legal. Queue is not in the same package nor is it a subclass of ListNode.