Name _____

The following might be names of patterns: Abstract Class, Abstract Factory, Adapter, Bridge, Builder, Chain of Responsibility, Client, Collaborator, Command, Command Processor, Composite, Context, Decorator, Façade, Factory Method, Flyweight, Interpreter, Mediator, Memento, Observer, Prototype, Proxy, Singleton, Specification, State, Strategy, Template Method, Visitor.

1. (3 points) What design pattern would you use to make it easy to change the implementation of an abstraction?

2. (3 points) What design pattern should you think of when you want to hide how you construct a complex object?

3. (3 points) What design pattern should you think of when when you wish to send a method to one of several objects without specifying the receiver explicitly?

4. (3 points) What design pattern would you use when you have a group of related objects that are designed to work together and you need to insure that they are used together?

5. (3 points) Which design pattern would you use when you want a client to create a new object without explicitly specifying the class of the new object?

6. (3 points) What design pattern would you use to make it easy to change the algorithm a class uses?

7. (3 points) What design pattern would you use when more that one object may handle a request and you don't know in advance which object will handle a particular request?

8. (3 points) What design pattern might you use when you wish to reduce tight coupling between classes?

9. (3 points) What design pattern would you use when you only have the binary of a class and need to modify the signature of some of the methods in the class?

10. (3 points) What design pattern would you use to reduce dependence on hardware and/or software platforms?

11. (10 points) What is the McCabe Cyclomatic complexity. What does it tell us about test cases.

12. (10 points) Explain how the Bridge pattern works.

13. (10 points) Both the Memento and Command patterns can be used to implement undo.

    a. Give one situation or condition where the Memento should be used.

    b. Give a situation or condition unrelated to the one in part a where the Memento should be avoided.

14. (10 points) Design patterns have consequences, some good and some bad.

    a. Give one good consequence of the builder pattern.

    b. Give one bad (or negative) consequence of the builder pattern.

15. (10 points) The following methods are from classes A & B that have a common superclass C. Modify (that is refactor) the methods using Template method and Factory Method. Show all the methods you would create and which classes they would be in.

```
Class A
send( Request aRequest, URL anAddress) {
        log( “send in class A called”);
        NetClient sender = new HttpClient().
        sender address(anAddress).
        Repsonse result = sender process( aRequest).
        if (result is404()) {log(“Error in sending message”)};
}

Class B
send( Request aRequest, URL anAddress) {
        log( “send in class B called”);
        NetClient sender = new FtpClient ().
        sender address(anAddress).
        Repsonse result = sender process( aRequest).
        if (result isEmpty ()) {log(“Error in sending message”)};
}
```

16. (10 points) The text claims that Mediator and Observer are competing patterns. Explain how they compete with each other. What are the strengths of each pattern over the other.

17. (10 points) Suppose there is a class Foo with the following method:

```
Object doSomething(Object x) {
        if (flag == "first")      return doFirst(x);
        if (flag == "second" ) return doSecond(x);
        if (flag == "third" ) return doThird(x);
        throw new Exception("Illegal option");
}
```

    This looks like a case statement. This might be OK if flag came from user input, but you discover that it is assigned in several methods and is always one of the string literals you see above. You notice that the doFirst() method calls a private method removeAll() of the foo class and is the only place where removeAll() is called. You notice that all three call a private method setUp. You decide to get rid of the case statement by using the Strategy pattern. Describe in detail what you would do. Show the new implementation of doSomething() for class Foo.