CSIE 5905004 **Name:** _____
Fall 2022
Final
1/9/2023, 1:10 pm - 2:40 pm
Time Limit: 90 Minutes **ID:** _____

This exam contains 6 pages (including this cover page) and 6 problems. Check to see if any pages are missing. Enter all requested information on the top of this page, and put your ID on the top of every page, in case the pages become separated.

You may use only the following materials: GoF textbook and the SOLID papers on this exam. You may *not* use your phones, laptops or any electronic devices on this exam.

You are required to show your work on each problem on this exam. The following rules apply:

- **Organize your work**, in a reasonably neat and coherent way, in the space provided. Work scattered all over the page without a clear ordering will receive very little credit.
- **Mysterious or unsupported answers will not receive full credit**. A correct answer unsupported by explanations will receive no credit; an incorrect answer supported by substantially correct explanations might still receive partial credit.
- If you need more space, use the back of the pages; clearly indicate when you have done this.

Do not write in the table to the right.

Problem	Points	Score
1	20	
2	20	
3	20	
4	20	
5	10	
6	10	
Total:	100	

1. This problem is about *composition* and *inheritance*.

   (a) (10 points) *True or False*: a design pattern of the object scope uses composition but not inheritance. If true, why? If false, give a counter-example.

   (b) (10 points) *True or False*: of the two versions of the Adapter (139) pattern, the class adapter is more flexible than the object adapter. Explain your answer.

2. This problem is about the *Liskov Substitution Principle*. Consider a base class `Base` with a data member `r` of the type `double` and a virtual method `sqrt()` for computing square root of `r`. When called, `sqrt()` checks if `r` is greater than or equal to 0. If so, it performs computation on `r`; if not, it throws an exception.

   In Johnny's application, his number `r` must be between 3 and 9. Since Johnny still needs the `sqrt()` member function, he writes a class `Derived` which publicly inherits `Base`. He overrides the method `sqrt()` with the following behavior: it checks if `r` is between 3 and 9. If so, it performs computation on `r`; if not, it throws an exception.

   (a) (10 points) *True or False*: the class `Derived` is compliant with LSP. Explain your answer.

   (b) (10 points) Write a unit test for checking compliance of LSP for `Base` and `Derived`.

3. In the *Open-Closed Principle*, it is said that "Since closure cannot be complete, it must be strategic".

   (a) (10 points) Early in our course this semester, Composite (163) is applied for relating the classes `Shape`, `Circle`, `CompositeShape`, etc. With regard to these classes, what kind of change are they strategically closed against? Under what kind of change does the strategic closure fail? What is the consequence?

   (b) (5 points) Among the 23 patterns in the textbook, which pattern can be applied to fix the failure of the strategic closure in (a)? Briefly explain how this is done.

   (c) (5 points) What is the new strategic closure after you apply the pattern? When does this new closure fail? What is the consequence?

4. In the `Shape` example in class, to read shape information stored on file and create corresponding shapes and their compositions in memory, we used three collaborating classes: `Parser`, `Scanner`, and `ShapeBuilder`. We wrote the constructor as in line 3 - line 6 shown below.

```
1  class Parser{
2  public:
3      Parser(std::string input):_input(input){
4          _builder = ShapeBuilder::instance();
5          _scanner = new Scanner(_input);
6      }
7      void parse(){ ... }
8      std::vector<Shape*> getResult(){ return _result;}
9  private:
10     std::string _input;
11     std::vector<Shape*> _result;
12     ShapeBuilder * _builder;
13     Scanner * _scanner;
14 };
```

   (a) (10 points) Argue that the constructor in line 3 - line 6 is a bad design by identifying the SOLID principle it violates.

   (b) (10 points) Suggest a fix to improve it. You can change the signature of the constructor in your design. Argue that your fix is a better design.

5. (10 points) Your team is developing a new application for your company. The new application must use a complicated existing system maintained by the other team. Luckily, you only need a service from the existing system, which involves several objects collaborating to achieve. The other team has been ordered to help your team by adding any object to the existing system for you. Suppose you were to ask the other team to help by adding a new object according to one of 23 patterns of the textbook, what pattern would you pick? Explain why.

6. (10 points) Use one of the `Shape` classes to give an example in which the Proxy (207) pattern is implemented exactly like the Decorator (175) pattern.