This test is open-book, open-note, closed-neighbor. There are 100 points total, and you will be given 50 minutes.

Part I: True or false, with explanations (40 points total)

This part comprises 10 statements. You should mark each statement as true or false; in addition, you should provide a one or two line justification of your answer. The true/false part of each statement is worth one point, while the justification is worth three points. (It is possible to get full credit for the justification even if you miss the true/false part.)

1. True or False. The most important reason to write down requirements for a project is because it lays the foundation for its design and implementation.

2. True or False. A layered design is especially useful when it is important to be able to predict the performance characteristics of a system.

3. True or False. A test suite that provides path coverage necessarily provides statement coverage.

4. True or False. Appropriately applying a software engineering technology (such as a testing technique, a programming language, a design technique, etc.) should be at least partially dependent on the underlying economics and cost-benefit analysis.

5. True or False. The best way to improve cohesion is by reducing coupling.

6. True or False. Layered designs help in the testing process.

7. True or False. Verification and validation are different names for the same activity.

8. True or False. Information hiding and design patterns were both developed because their originators had seen many instances of the approach in practice.

9. True or False. Design patterns are generally not an effective tool for helping design a system architecture.

10. True or False. In principle, both decompositions of KWIC described in Parnas' paper could be used to produce identical executables.

YOUR NAME: _____

Part II: Short answer (30 points total)

This part comprises six questions. You should answer each question in about a paragraph. Each question is worth five points.
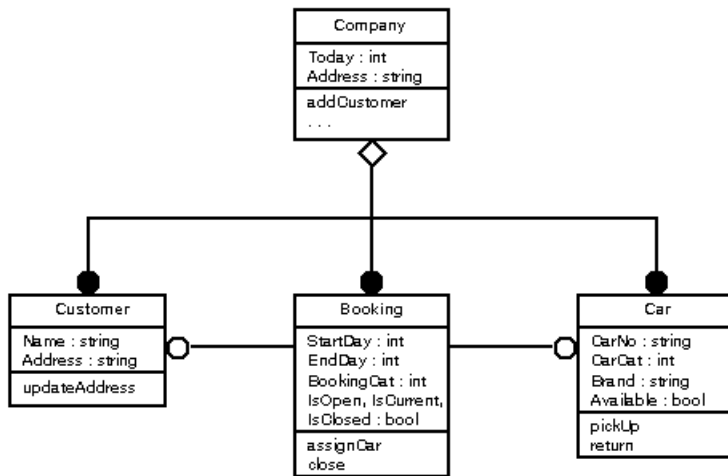
1. Briefly describe the fundamental distinction between the spiral model and the waterfall model.

2. To what degree, if any, do the kinds of relationships among program components (e.g., classes, functions, modules) depend on the programming language in which a program is written?

3. Briefly describe why the simple distinction that requirements capture the "what" and the implementation captures the "how" is naïve.

4. List two reasons why mathematically proving the correctness of programs has not become common in practice.

5. Briefly describe how Parnas' notion of program families relates to stepwise refinement.

6. Show a small example of an object-oriented class interface in which there are no public instance variables but it is still a terrible example of information hiding.

YOUR NAME: _____

Part III: Requirements Critique (20 points total)

Consider the following object model that is intended to represent part of the requirements for a car rental system.

Object Model

```
                    ┌──────────────────┐
                    │     Company      │
                    ├──────────────────┤
                    │ Today : int      │
                    │ Address : string │
                    ├──────────────────┤
                    │ addCustomer      │
                    │ . . .            │
                    └──────────────────┘
                             ◇
            ┌────────────────┼────────────────┐
            ●                ●                 ●
   ┌─────────────────┐ ┌──────────────────┐ ┌──────────────────┐
   │    Customer     │ │     Booking      │ │       Car        │
   ├─────────────────┤ ├──────────────────┤ ├──────────────────┤
   │ Name : string   │○│ StartDay : int   │○│ CarNo : string   │
   │ Address : string│ │ EndDay : int     │ │ CarCat : int     │
   ├─────────────────┤ │ BookingCat : int │ │ Brand : string   │
   │ updateAddress   │ │ IsOpen, IsCurrent,│ │ Available : bool │
   └─────────────────┘ │ IsClosed : bool  │ ├──────────────────┤
                       ├──────────────────┤ │ pickUp           │
                       │ assignCar        │ │ return           │
                       │ close            │ └──────────────────┘
                       └──────────────────┘
```

Critique this as part of a system requirement description. (In this case, the open diamond means aggregation, the filled-in circle means many-to-one, and the open circle means one-to-one.) In particular, identify four specific strengths and/or weaknesses of this description. (You must include at least one strength and at least one weakness.)

YOUR NAME: _____

Part IV: Information Hiding Critique (10 points total)

A fundamental premise of information hiding is that a module should only expose its functionality, leaving the clients are unaware of how the module is implemented. The intent, of course, is to allow the implementation to be changed without changing the clients.   One research group from Xerox PARC claims that this is a naïve view of software design, because clients often depend on the performance characteristics of a module, which is determined by the implementation. In particular, different clients may simultaneously have different performance needs. (For example, a client that draws hundreds of small windows on the screen might require a different implementation from a client that draws at most 10 large windows on the screen.)

Is this criticism of information hiding justified?   If so, why? If not, why not?

YOUR NAME: _____