CSE/IT 326: Software Engineering
Spring 2021

Final Exam, May 4th, 2021

Your Name: Marisa Lorgas **Student ID:** 900332477

Instruction: *This is a take-home exam, and following is the general instruction for the exam. Students are expected to follow this instruction, and failing to do so may result in a grade reduction or an Honor Code violation.*

1. *You can work on the exam for 24 hours. Please print out the exam on 8.5" x 11" paper, single sided, and complete the exam.*
2. *To submit your exam answer, submit your finished exam as a* ***PDF*** *to CANVAS by* ***May 4th (Tuesday), 11:59pm.***
3. *You may refer to the lecture notes, homework, and textbook/print. You are not allowed to refer to any other material.*
4. *You may not consult with any other person regarding the exam. You may not check your exam answers with any person.*

	POINTS	YOUR SCORES
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	10	
Total	100	

1. **[Software Process Models, 10 points]** Answer the following questions.

   a. Give the correct order of activities in the following software design workflow. **[2 points]**

A.	*Subsystem design*	3rd
B.	*Object design*	4th
C.	*Architecture design*	2nd
D.	*Use case design*	1st

   b. Briefly discuss Capability Maturity Model in terms of ***i***) when a software development process is defined to be *mature*, and ***ii***) what are the five levels in the model. **[4 points]**

   i) A software development process is mature if the development activities are well defined and if management has some control over the quality, budget and schedule of the project.

   ii) Initial level, Repeatable level, Defined level, managed level and Optimizing level

   c. Briefly discuss the software development lifecycle (SDLC) model adopted for your final project of this class. Is it a sequential or iterative approach? Provide at least two advantages of the SDLC model your team has used. **[4 points]**

   Our final project followed the Unified process model. Overall it was an sequential approach, with some of the phases being iterative, mostly the Construction phase.
   Advantages to sequential approach:
   - Nice milestones
   - No need to look back (linear system)
   - Easy to check progress

2. **[Modeling with UML, 10 points]** Modeling is to build an abstraction of reality.

   a. Consider a Banner system such as NMT Banweb. Identify at least **four** different actors that interact with this system. **[2 points]**

   - Student
   - Employee
   - Administrator
   - Professor

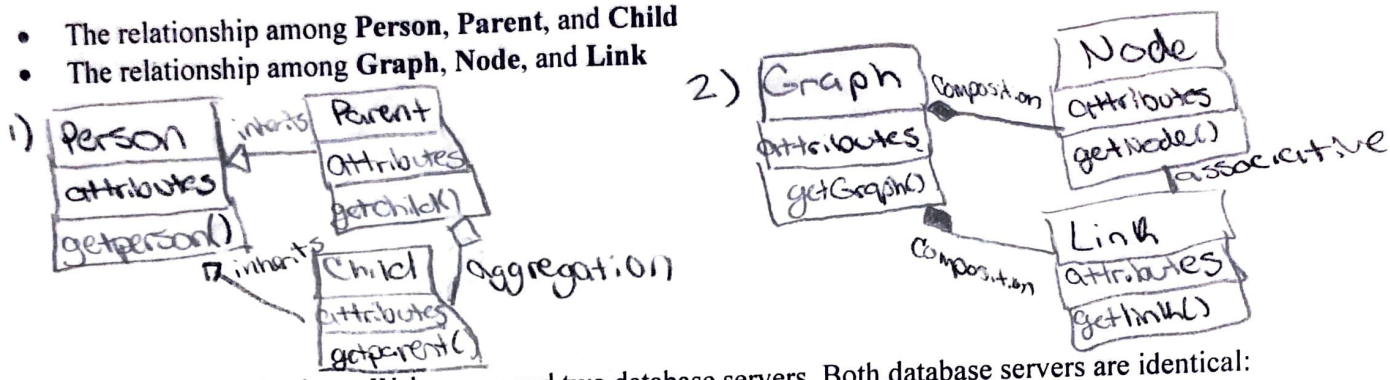
   b. What is the difference between a scenario and a use case? When do you use each construct? **[2 points]**

   Scenario → A concrete, focused, informal description of a single feature of the system used by a single actor. Application Domain situation

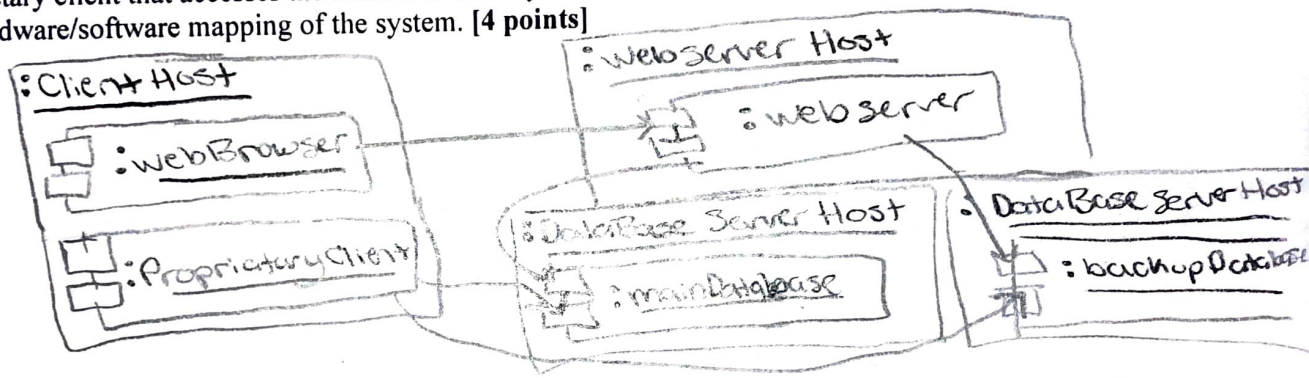
   Use Cases: Abstraction that describes a class of scenarios Solution Domain situation

c. Use class diagram to model the following. **[2 points]**

- The relationship among **Person**, **Parent**, and **Child**
- The relationship among **Graph**, **Node**, and **Link**



d. Consider a system that has a Web server and two database servers. Both database servers are identical: the first acts as a main server, and the second acts as a redundant backup in case the first one fails. Users use Web browsers to access data through the Web server. They also have the option of using a proprietary client that accesses the databases directly. Draw a UML deployment diagram representing the hardware/software mapping of the system. **[4 points]**

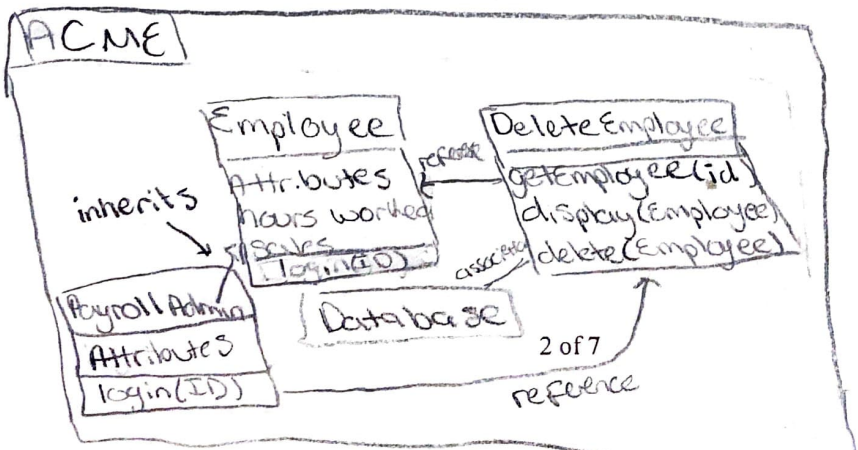


3. **[Requirement Elicitation and Analysis, 10 points]** As the head of Information Technology at Acme, Inc., you are tasked with building a new payroll system to replace the existing system that is hopelessly out of date. Acme needs a new system to allow employees to record time card information electronically and automatically generate paychecks based on the number of hours worked and total amount of sales (for commissioned employees). Following is the Flow of Events of the **Delete an Employee use case** of the payroll system:

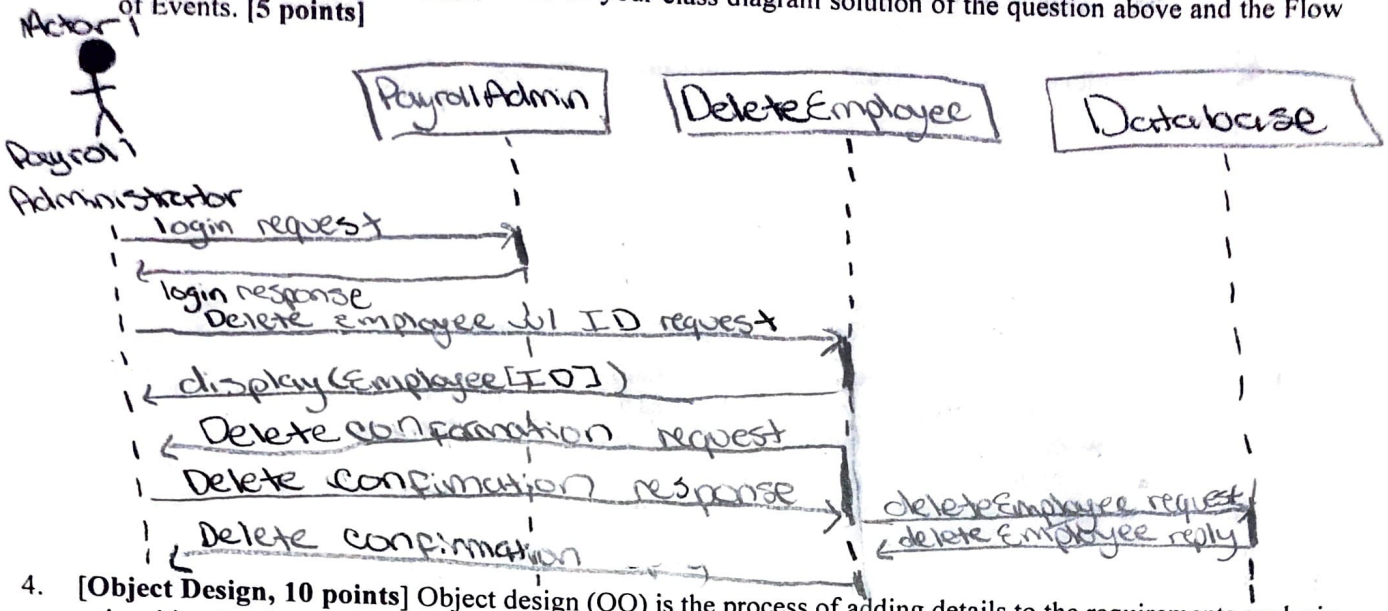
Flow of Events

1. The system requests that the Payroll Administrator specify the employee id.
2. The Payroll Administrator enters the employee ID. The system retrieves and displays the employee information.
3. The system prompts the Payroll Administrator to confirm the deletion of the employee.
4. The Payroll Administrator verifies the deletion.
5. The system removes the employee from its database.

a. Construct a class diagram using the Flow of Events. **[5 points]**



b. Construct a sequence diagram based on your class diagram solution of the question above and the Flow of Events. **[5 points]**



4. **[Object Design, 10 points]** Object design (OO) is the process of adding details to the requirements analysis and making implementation decisions.

4) Object model optimization: Transforms the object design model to address performance criteria such as response time.

a. Briefly discuss the four key activities for object design. **[2 points]**

1) Identification of existing solutions (reusability): - use of inheritance - Design patterns - Off the shelf components and additional solution objects
2) Interface specification: Describes precisely each class interface
3) Object model restructuring: transforms the object design model to improve its understandability and extensibility

b. Briefly discuss the OOD principle of "*Liskov Substitution*", along with its example and benefits. **[4 points]**

A Basic principle of OOP and OOD → LSP or Liskov substitution principle says the following: if S is a subtype T, then object of type T may be replaced with objects of types S without altering any of the desirable properties of the program

would fail if we just had a Bird class to inherit from

Ex: public class Bird {} public class FlyingBird extends Bird {} public class Robin extends FlyingBird {} public class Penguine extends Bird {}

Benefits: Allows for abstraction and enables a hierarchy

c. Briefly discuss the OOD principle of "*Dependency Inversion*", along with its example and benefits. **[4 points]**

Another SOLID Principle of design is the Dependency Inversion Principle (DIP) and it depends on abstractions, not concretions

Ex. Lets say we have a calculator class with an add and subtract function. If we wanted to add a new operation to our calculator, we cant just add it to the class. Instead make a Calculator class w/ an operation Attribute, where any operation can be called. Operation classes will be their own classes

Benefits: Reduces Dependency and makes code more reusable and reusable

5. **[Design Patterns I, 10 points]** Answer the following questions.

a. Briefly discuss what design patterns are. **[2 points]**

A design pattern describes a problem which occurs over and over again in our environment, then it describes the core solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same twice.

b. List two situations where it would be appropriate to apply the **Decorator** design pattern. **[4 points]**

Appropriate situations to apply Decorator design patterns:

1) To add responsibility to individual objects dynamically and transparently, that is, without affecting other objects

2) for responsibility that can be withdrawn 3) when extension by subclassing is impractical

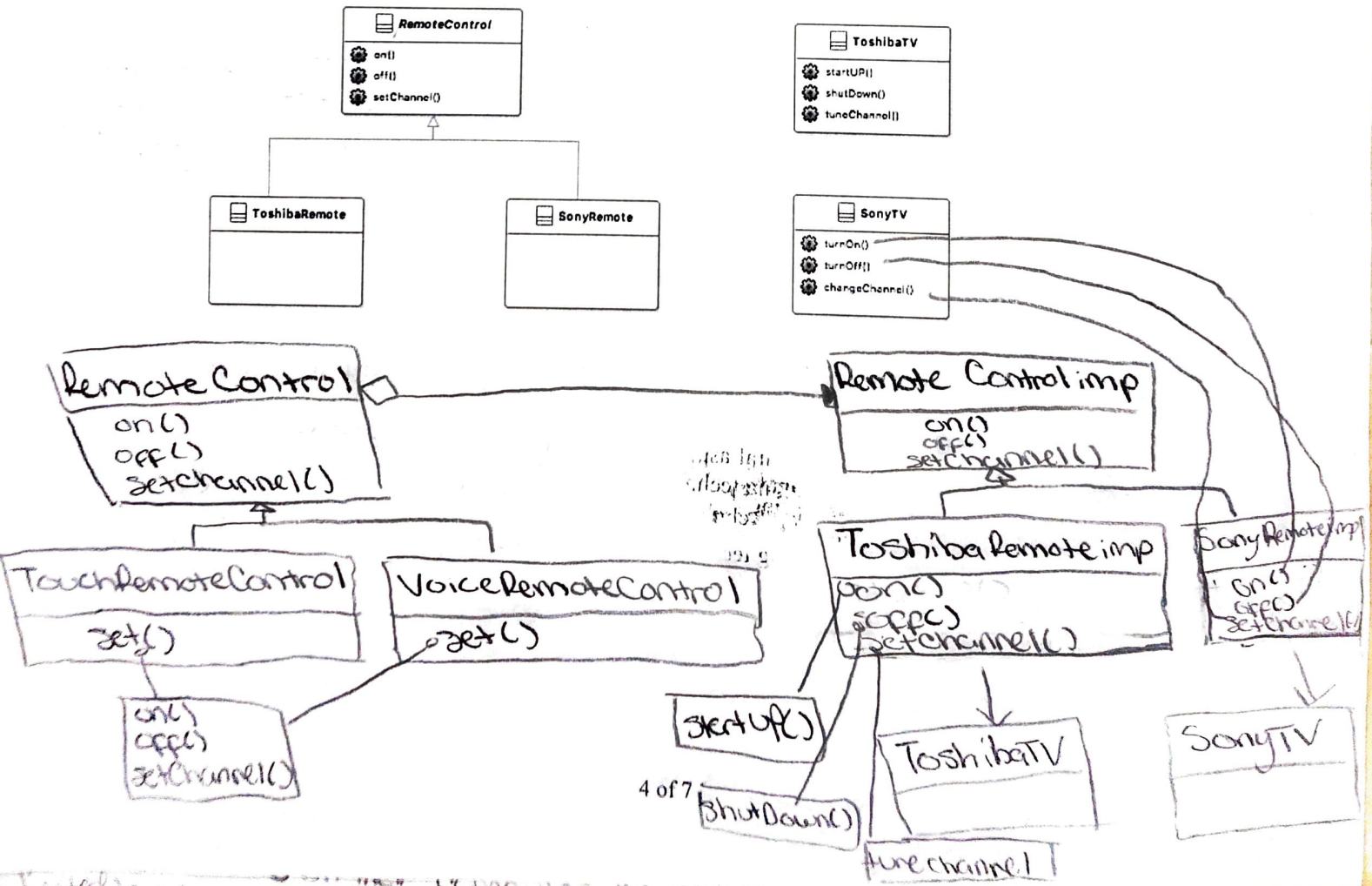
c. List two consequences of using the **Abstract Factory** design pattern. **[4 points]**

1) Supporting new kinds of products is difficult, because it promotes consistency among products

2) It isolates concrete classes; since a factory encapsulates the responsibility and process of creating product objects, it isolates clients from implementation classes

6. **[Design Patterns II, 10 points]** Answer the following.

a. Modify the following design using the **Bridge** design pattern so that you can vary the implementation over two TVs as well as the interface (*RemoteControl*). Use two new concrete classes, **TouchRemoteControl** and **VoiceRemoteControl**, as variations of the interface. **[5 points]**



Voice remote control is made exactly the same but with the VoiceRemote class name (can't run out of room)

b. Transform your class diagram solution of the question above into Java (or pseudo) code. **[5 points]**

```
abstract class RemoteControl {
    private RemoteControlImp imp;
    public void on() {...}
    public void off() {...}
    public void setChannel() {...}
}

class TouchRemoteControl extends RemoteControl {
    public void setRemoteControlImp(RemoteControlImp imp);
    public void set() {
        on();
        off();
    } setChannel();
}
```

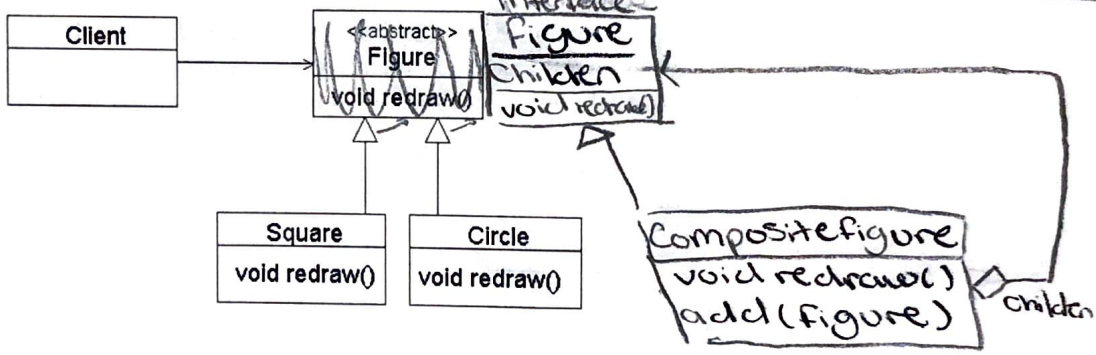
```
Interface RemoteControlImp {
    public void on();
    public void off();
    public void setChannel();
}

public class ToshibaRemoteImp
    implements RemoteControlImp {
    public void on() {
        startUP();}
    public void off() {
        shutDown();}
    public void setChannel() {
        tuneChannel();}}

public class SonyRemoteImp
    implements RemoteControlImp {
    public void on() {turnOn();}
    public void off() {turnOff();}
    public void setChannel() {changeCha
        nnel();}
}
```

7. **[Design Patterns III, 10 points]** Answer the following questions.

a. Modify the following design using the **Composite** design pattern so that a Client can interact transparently with either a single Figure or a group of Figures. **[5 points]**



b. Transform your class diagram solution of the question above into Java (or pseudo) code. **[5 points]**

```
Interface Figure {
    public children;
    public void redraw();
}

public Square implements Figure {
    public void redraw() {...}
}

public Circle implements Figure {
    public void redraw() {...}
}
```

```
public CompositeFigure implements Figure {
    public void redraw() {...}
    public void add(Figure F) {...}
}
```

8. **[OCL and Mapping Models to Codes, 10 points]** Answer the following questions.

a. Consider the `Stack` class in the `java.util` package for a last-in-first-out (LIFO) stack of objects. Write post conditions in OCL for the following operations: **[6 points]**

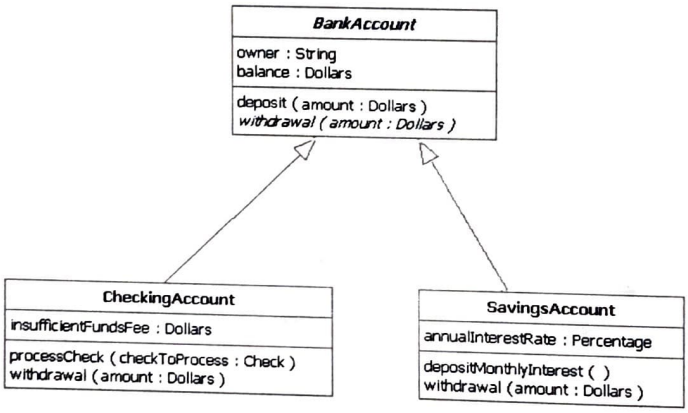
i. `E pop()` removes the object at the top of this stack and returns that object
ii. `E push(E item)` pushes an item onto the top of this stack
iii. `E peek()` looks at the object at the top of this stack without removing it from the stack

i) Context Stack::pop()
post: self.size = previous.size - 1

ii) Context Stack::push(E item)
post: self.peek() == item
post: self.size = previous.size + 1
post: self[self.size] == item

iii) Context Stack::peek
post: No post condition

b. Map the following inheritance relationship among BankAccount, CheckingAccount, and SavingAccount to relational tables using vertical mapping. **[4 points]**



type is either Checking Account or Savings Account

BankAccount Table

ID: long	owner: text[32]	balance: Dollars	...	type

CheckingAccount Table

ID: long	insufficientFundsFee: Dollars	...

SavingsAccount Table

ID: long	annualInterestRate: Percentage	...

9. **[Software Security, 10 points]** Answer the following questions.

a. Briefly define what the least privilege principle is and discuss also at least one example of security ramification when violated. **[4 points]**

Least privilege principle → one of the fundamental security principl
- Perform tasks with the least set of priviledges required
- Use elevated privileges for the shortest possible times
- Minimize potential damage

Security Ramification Ex: Lets say a software gives priviledges to an emplo to add to the systems database. If this employee were to get some kind of phishing or trojan virus on their computer, the malicious attacks on the system would be limited to adding to the database, however if this employee has root access (breaking the principle) then the attacks can spread throughout

b. Cryptography is one of the fundamental security techniques to protect communications and data used by applications. Consider you are going to develop an online banking system that will allow customers to check their account balance and transfer fund. Instead of using AES algorithm, considering performance, your team member proposed to use a much faster encryption algorithm based on XOR as follows:

```
void EncryptData(char *szKey, DWORD dwKeyLen,
                 char *szData, DWORD dwDataLen){
    for (int i = 0; i < dwDataLen; i++) {
        szData[i] ^= szKey[i % dwKeyLen];
    }
}
```

Discuss why this is a bad idea. **[6 points]**

Using this XOR algorithm leaves a vulnerability in the system, where its very simple to identify someones password in a datafile or binary file. Because ASE algorithm uses different keys to encrypt and decrypt files in block chunks, it does not have this same problem.

10. **[Software Testing, 10 points]** Answer the following questions.

c. Briefly define the <u>difference</u> between validation and verification. **[3 points]**

Validation → the process of evaluating software at the end of the software development, to ensure compliance w/ intended usage

Verification → the process of determining whether the product of a given phase of the software development process fulfills the requirement established during the previous phase.

d. Briefly discuss the four <u>testing steps</u>. **[4 points]**

- Acceptance Testing: Determines if a system is ready to be released
- System Testing: when entire application is tested
- Integration Testing: finds interface defect
- Unit Testing: for a function, procedure, or individual program.

e. White-box testing focuses on structural aspects; black-box test focuses on the functional requirements. Identify at least <u>three</u> white-box testing techniques. **[3 points]**

White-box testing focus: thoroughness (coverage). Every Statement in the component is executed at least once.

4 types of white-box testing:

- Statement Testing
- Loop testing
- Path Testing
- Branch testing