SOFTWARE IMPLEMENTATION

INTRO TO CSCU9P6

1. What is the **V model**? Draw a diagram of it.
2. Describe some examples of **detailed design activities**.
3. What is involved in **refining associations**?
4. Describe how **dependencies** work.
5. What is an **attribute-based association**?
6. How can we implement **associations**?
7. Give some **examples** of associations we can have.
8. What is an **association class**? How can we implement them?
9. What is **aggregation**, and how do we implement it?
10. What is **composition**, and how do we implement it?

USE CASE & SEQUENCE DIAGRAMS

1. What is a **use case**?
2. What is a **sequence diagram**?
3. Describe how **use cases** and **sequence diagrams** relate to one another, and how they are both implemented.
4. How can we show **actions/boundary classes** in sequence diagrams?

IMPLEMENTING STATE DIAGRAMS

1. What is a **sequence diagram**? How does it work?
2. Describe what is meant by the following terms:
    a. **Event name**
    b. **Event arguments**
    c. **Guard condition**
3. Draw an example of a **state diagram**.
4. What is an **implementation requirement**?
5. If a state diagram has an **action** on **transition** from **start symbol** to an **initial state**, where is that action placed?

IMPLEMENTATION ISSUES – REFACTORING ON DESIGNS

1. What is meant by **refactoring**?
2. Describe some examples of **low level** and **higher-level** refactoring operations.
3. Why might we want to split a complex class into two or more **independent** classes?
4. How else can we use **class splitting**?
5. Why might we want to use refactoring to increase **reuse** within a system? How would we do this?

DESIGN PATTERNS

1. Why might we want to **reuse expressions**?
2. What is a **design pattern**?
3. Describe the **3 main types** of design pattern.
4. What is the **composite pa**ttern? Describe how it works.
5. What is the **publisher-subscriber pattern**? Describe how it works.

MVC & IMPLEMENTATION ISSUES

1. What is the **MVC architecture**? Describe in detail how it is implemented.
2. How do the Java library classes **Observer** and **Observable** work in terms of the MVC architecture?
3. Why might we want each controller to appear in a **separate** window?

SOFTWARE TESTING

1. Describe what is meant by the following types of tests:
    a. **Unit/component** test
    b. **Integration** test
    c. **System** test
    d. **Validation/acceptance** test
2. What is the difference between **verification** and **validation**?
3. Why is **perfect testing** impossible?
4. Describe the **testing process** involved in the 4 types of tests mentioned above.
5. Describe what is involved in the **formality of testing**.
6. Why should we bother with **documentation**?
7. What is the difference between **test data** and **test cases**?
8. Describe the difference between **black box** and **white box** testing.
9. Draw a simple diagram of each in terms of **how** testing works on them.
10. Can unit testing be both **white** and **black box testing**?

DEBUGGING

1. Describe the steps involved in the **general debugging process**.
2. Why is **retesting** the program important?
3. Describe some approaches to locating a **fault**.
4. Describe some **diagnostic statements** we can use.
5. What is **interactive debugging**? How does it work?
6. What is a **breakpoint**? How do they work?
7. Describe the **JVM scheme**.

INTEGRATION TESTING

1. What is **integration testing**?
2. Should integration testing be **black** or **white box testing**?
3. What is the main **difficulty** with **integration testing**? How can we address this?
4. Describe what is meant by **top-down** and **bottom-up** testing and draw a simple diagram of how both works.
5. What is meant by **alpha** and **beta testing**?
6. Name some **other** forms of testing.
7. What is **stress testing**? How does it work?

JUNIT

1. What is **JUnit**?
2. Describe some of the **features** it provides.
3. Describe the process of **writing** and **performing** a JUnit test.
4. What is a **fixture**? How are they used in JUnit testing?
5. What **other features** might a test class contain?
6. What is a **test suite**?

7. Describe what is meant by **test-driven development**.

CONFIGURATION MANAGEMENT

1. Describe what is meant by **configuration management**.
2. Why might there be **parallel versions** of components?
3. Why might there be **sequentially related versions** of components?
4. What is a **configuration database**? How can it be used?
5. What is a **CASE tool**?
6. What is a **build control tool**?
7. Name the **three main approaches** and **describe** how they work.

VERSION CONTROL & COLLABORATIVE WORKING

1. What is meant by **version control**?
2. Understand and describe the **version numbering scheme** i.e. what does X.Y.Z mean?
3. What is meant by the following terms?
    a. **Branch**
    b. **Fork**
    c. **Commit**
    d. **Push**
    e. **Checkout**
4. How does **version management** work?
5. What is a **repository**?
6. How does a **VCS tool** work?
7. What is a **delta**?
8. Name some **well-known** VCS tools.

SOFTWARE ENGINEERING MATHEMATICS & SPECIFICATION

SOFTWARE ENGINEERING SPECIFICATION/MATHEMATICS

1. What is **formal specification**?
2. Why **use** formal specification?
3. Describe some **advantages** of formal specification.

ALLOY OVERVIEW

1. Describe what is meant by the following terms:
    a. **Set**
    b. **Relation**
    c. **First-order logic**
2. What is the **purpose** of Alloy?
3. What is the **Alloy Analyser**?
4. Describe what **sig**, **pred** and **run** mean in Alloy.
5. What is a **model-based notation**?

BASIC SET THEORY IN ALLOY

1. What is a meant by **univ**, **none** and **int** when referring to sets?
2. Describe how we can **add our own sets** in Alloy.
3. What is a **multiplicity**? Give examples of how they can be used in Alloy, and which ones there are.

4. How can we create **subsets** of a set in Alloy?
5. What is an **abstract signature**?
6. Describe some of the **operations** we can perform on sets (union, difference, etc.)
7. What **logical expressions** can we use with sets? Name and describe some examples.
8. Give some examples of **general**, **associative**, **commutative** and **distributive laws** in set operations.
9. How might a signature contain **fields**?
10. How can a signature be given a **constraint**?
11. What is a **predicate**? How can we use them in Alloy?
12. What is a **fact**? How can we use them in Alloy?
13. What is an **assertion**? How can we use them in Alloy?

RELATIONS & RELATIONAL OPERATORS

1. What is an **atom**? Describe some properties an atom has.
2. What is a **relation**? How are they **constructed**? Draw an example of them, making sure to specify what **tuple** and **arity** mean.
3. What is meant by the following terms when referring to relations:
    a. **Unary**
    b. **Binary**
    c. **Ternary**
    d. **Multirelation**
    e. **Scalar**
4. Describe the difference between a **function** and **injective relation**. Draw examples of each **combination**.
5. Describe the difference between a **total** and **subjective relation**. Draw examples of each **combination**.
6. Show how we can represent **multiplicities** in relations.
7. Describe the difference between **domain** and **range**.
8. What is the **identity relation**?

MORE ON RELATIONS/RELATIONAL OPERATORS

1. Name and draw all the **relational operations** that can be used in Alloy.
2. For all the above relational operations, provide an **example** of how they can be used.
3. What is meant by **transitive closure**?

MATHEMATICAL LOGIC

1. Name and describe all the **mathematical operators**, as well as how they are represented in Alloy.
2. Give an **example** of how each operator works.

MORE ON LOGIC

1. List some examples of **equivalent logical expressions**.
2. What is a **quantified expression**?
3. Give some examples of **quantified expressions** in Alloy.

MORE ON LOGIC (ELECTRIC BOOGALOO)

1. How can we express **cardinality** in Alloy?
2. What is a **let expression**? Give an example of how it can be used in Alloy.
3. What is a **comprehension**? How can we perform them in Alloy?

MODEL CHECKING

1. What is meant by **first order logic**?
2. What is **model checking**?
3. What is the **signature** of a **specification**?
4. What is the **structure** of a **signature**?
5. What is the **model** of a **specification**?
6. Describe the difference between **consistency** and **validity**.

MORE ON MODEL CHECKING

1. Describe the two main **model checking commands** in Alloy and how they work.
2. What is meant by **scope**?
3. Describe some of the **limitations** of model checking in Alloy.
4. What is meant by the "**small scope**" hypothesis?

DYNAMIC SYSTEMS

1. Why is it difficult to model **dynamic systems** in Alloy?
2. How can we get **around** this?
3. Describe an example of a **dynamic system**, and how it would be **represented** in Alloy.
4. What do we need to know to specify the **state** of the system?
5. What do we need to ask to be able to specify a **static operation**?
6. What do we need to ask to be able to specify a **dynamic operation**?
7. What is a **sanity check**? How can it be **performed**?

PROJECT MANAGEMENT & QUALITY ASSURANCE

COST & EFFORT ESTIMATION

1. Explain why it is important to be able to estimate the **effort** and **cost** that a software project will involve.
2. Discuss what the **difficulties** are in making such estimates.
3. Describe two methods that have been proposed for **project cost estimation**.
4. How would you go about estimating **cost** and **effort** for the case study projects?

ACTIVITY PLANNING

1. Explain the reasons for producing a **project activity plan**.
2. Discuss what is meant by an **"activity"** and how a project manager might go about identifying the different activities that make up a project.
3. Describe in detail the **general structure of a project activity plan**, the kinds of **diagrams** that it might contain, and the ways in which it might be **analysed**.
4. Identify activities and sketch a possible **activity plan** for the case study projects.

TEAM MANAGEMENT

1. Explain why **project team management** has been described as one of the most challenging aspects of **software project management**.
2. Discuss what **factors** should be considered in **selecting** and **training** staff to make up a new team.
3. Write a list of **guidelines** for project managers to follow when **managing teams**, giving reasons for each guideline.

4. Discuss how you would go about **selecting** and **managing** a team for the case study projects.

RISK MANAGEMENT

1. Explain why it is important to **foresee** and **manage risks** in software projects.
2. Discuss the kinds of **risks** that affect software projects.
3. Describe in detail techniques that project managers can use to **foresee risk**, **measure its impact**, and **monitor** and **mitigate** its **effects**.
4. Discuss how you would apply these **risk management techniques** to the case study projects.

QUALITY ASSURANCE

1. Discuss what is meant by **"quality"** in the context of software.
2. Describe some ways in which quality can be **measured**.
3. Describe **techniques** that have been proposed for software quality assurance.
4. Explain what approach you would use for **quality assurance** in the case study projects.

PROJECT MANAGEMENT TOOLS

1. Explain why **software tools** might be helpful in project management.
2. Describe in general the kinds of **functions** that a project management tool should provide.
3. Give a detailed description of at least **two specific project management tools**, explaining what functions they provide and discussing their **strengths** and **weaknesses**.
4. State whether you would recommend one of these **tools** for use in the case study projects, explaining your reasons.