The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, AUTUMN SEMESTER 2022-2023

DEVELOPING MAINTAINABLE SOFTWARE

Time allowed: **Sixty (60) Minutes (One Hour)**

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer All SIX questions

Total Marks Available: **50**

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so

INFORMATION FOR INVIGILATORS:
Collect both the exam papers and the answer booklets at the end of the exam.

SECTION A: Object-Oriented Concepts and Maintenance Principles

Question 1: The basic object-oriented programming (OOP) is based on the concept of abstraction, encapsulation, inheritance, polymorphism, and interface.

a) What are the differences between aggregation and composition in object-oriented programming?

[4 marks]

b) Describe the concept of abstraction and how it is useful in software maintenance.

[6 marks]

Question 2:

a) Explain the Open-Closed principle in OOP.

[6 marks]

b) Use the Open-Closed principle to refactor the Java code below.

[8 marks]

```
public double calculateSalary(Employee employee) {
  double salary = 0.0;
  if (employee is Developer) {
    Developer developer = (Developer) employee;
    salary += employee.workingHours * employee.hourlyRate;
  }
  else if (employee is TeamLeader) {
    TeamLeader teamLeader = (TeamLeader) employee;
    salary += teamLeader.workingHours * teamLeader.hourlyRate * teamLeader.bonus;
  }
  else if (employee is Recruiter) {
    Recruiter recruiter = (Recruiter) recruiter;
    salary += recruiter.workingHours * recruiter.hourlyRate * recruiter.bonus * 1.5;
  }

  return salary;
}
```

End of Section A: Total 24 marks

SECTION B: Design Principles and Refactoring

Question 4: Briefly explain the design principle "Favor composition over inheritance". Provide a design example that follows this design principle (You don't have to write code or provide diagrams. Verbal description is expected). Finally, state how this design principle relates to 'polymorphism' in object-oriented programming. Note: Your answer for this question should not exceed 100 words in total.

[6 marks]

Question 5: Describe the essential components that form a Decorator pattern.

[6 marks]

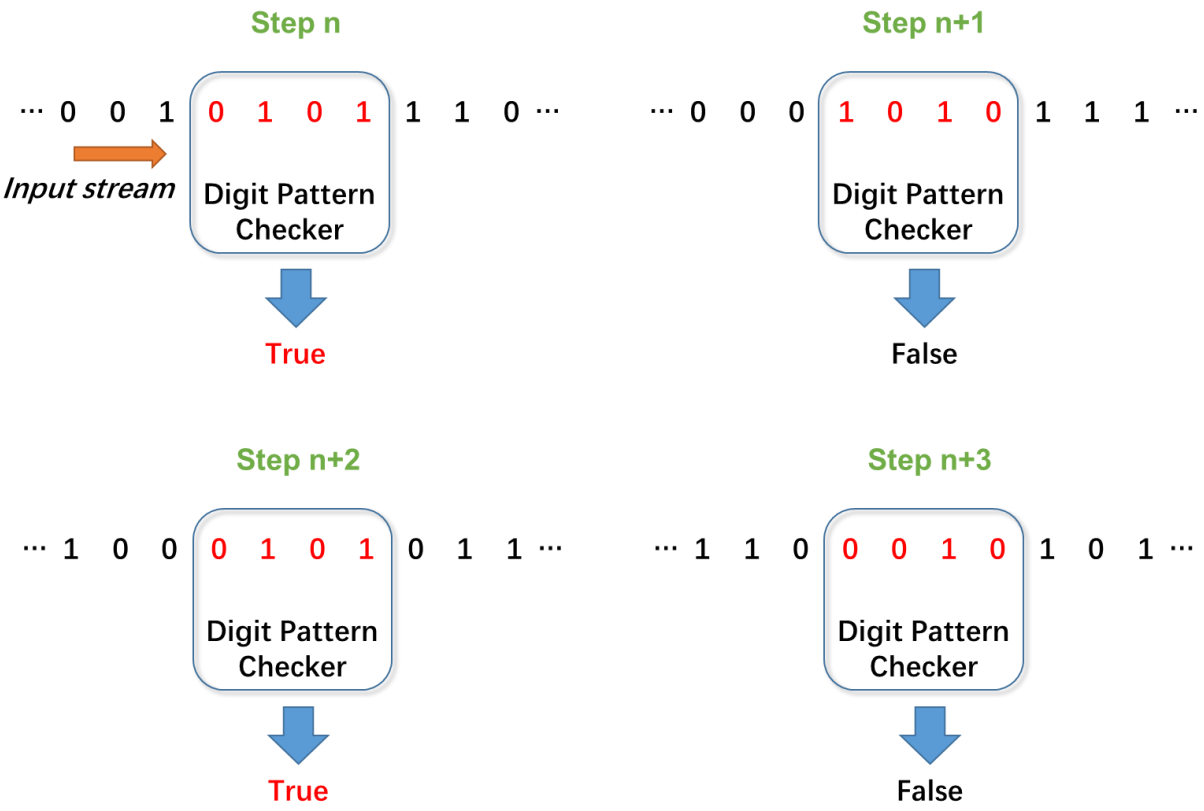
Question 6: A serial pattern checker is a system that keeps checking the serial digit pattern input as a stream into the system. The input stream is composed of serial random digits, namely 0s and 1s. The digit pattern is a fixed digit sequence, e.g., 0101. Once the system captures this pattern, it should output a positive signal, such as a Boolean 'true' to indicate that. At other times, it output a negative signal, such as a Boolean 'false' to indicate that the current sequence is not the pattern. The figure below shows an example of the system checking on pattern '0101'. Note that the digit stream flows from left to right. With the above description, answer the two questions below.

1. Design a state machine for the Digit Pattern Checker system to recognize the digit pattern '0101'. The state machine should contain exactly five (5) states. Draw the state machine, with properly designed states and transitions.

[10 marks]

2. Assume that you use Java to implement this system. Design your program with the State Pattern, by drawing the class diagram. Note: You are not required to write the code. Only the class diagram with State Pattern is expected.

[4 marks]



End of Section B: Total 26 marks