TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

ACADEMIC YEAR 2024/2025

MAY/JUNE EXAMINATION

BACS2083 FORMAL METHODS FOR SOFTWARE ENGINEERING

FRIDAY, 30 MAY 2025 TIME: 3.00 PM – 5.00 PM (2 HOURS)

BACHELOR OF SOFTWARE ENGINEERING (HONOURS)

Instructions to Candidates:

Answer **ALL** questions. All questions carry equal marks.

Question 1

a) Formal methods are considered as a branch within software engineering.

(i) In your own words, discuss *formal methods* in detail and explain their main goal. (5 marks)

(ii) Using an example of a scenario, describe how formal methods can be applied and beneficial in the software development lifecycle. (4 marks)

(iii) What is *model checking*, and how is it different from *theorem proving* in formal verification? (4 marks)

b) Using Z *axiomatic definition* notation, produce the relationships described below with appropriate Z data types. Provide an example of a set for each relationship.

(i) In a company, each employee is assigned a unique ID card, and no ID card is shared between employees. Produce a relationship between the employees and their ID cards. (3 marks)

(ii) In a university, each student is assigned to study a programme within a single faculty. Produce a relationship between the students and their faculties. (3 marks)

(iii) In a software company, each developer can work on multiple projects. Produce a relationship between the developers and their projects. (3 marks)

c) In Malaysia, every citizen aged 12 and above is required to obtain a unique identity card from the *Jabatan Pendaftaran Negara (JPN)*. Let [CITIZEN] represent the set of all citizens aged 12 and above, and [IC] represent the set of all registered identity card numbers in the system. The relationship between IC and CITIZEN can be represented in the following state:

Jpn

$$identityCard : CITIZEN \rightarrowtail\!\!\!\!\!\!\rightarrow IC$$

Interpret the relationship described in the state of *Jpn* above. (3 marks)

[Total: 25 marks]

Question 2

TechValley Sdn. Bhd. is a company that specialises in providing technical consulting and project management services. They often work with a large number of project-related documents such as technical specifications and client contracts. To ensure effective collaboration between the team members, TechValley has decided to implement a **Document Control System** (DCS) that allows for organised access and management of these documents.

Mr. Alex, the project manager at TechValley, has contacted your software development company to create a system that manages document check-out and check-in processes to ensure smooth and secure collaboration. You, as the system analyst, have gathered the following requirements from Mr. Alex and his team:

> When a user wants to edit a document, the DCS first checks if the user has the necessary permissions to make changes. If the user is authorised and no one else is currently editing the document, the system allows the user to check out the document. Once checked out, the document becomes locked, preventing others from making changes simultaneously. Once the document is checked out, the user can proceed with editing if they have read and write access.
>
> When the user completes their edits, they are required to check in the document. Checking in a document signifies that the user has finished editing and the document is now available for other users to check out and edit. Each time a document is checked in, the system creates a new version.

After studying further about the DCS, you have concluded to specify the system with the following information:

Basic types:

[USER] - the set of all possible users in the company
[DOCUMENT] - the set of all documents that can be shared and edited

Free types:

CATEGORY ::= technicalSpecification | designDocument | clientContract | internalReport
STATUS ::= locked | unlocked
ACCESS ::= readOnly | readWrite | admin

And a state space schema called *DocumentControl*:

DocumentControl

$$
\begin{aligned}
&documentId : \mathbb{P}\ DOCUMENT \\
&userId : \mathbb{P}\ USER \\
&documentTitle : DOCUMENT \rightarrowtail TITLE \\
&category : DOCUMENT \rightarrow CATEGORY \\
&permission : DOCUMENT \leftrightarrow USER \\
&checkedOutBy : DOCUMENT \rightarrow USER \\
&checkOutStatus : DOCUMENT \rightarrow STATUS \\
&versionHistory : DOCUMENT \rightarrow seq\ DOCUMENT \\
&accessLevel : USER \rightarrow ACCESS \\
\hline
&dom\ versionHistory = dom\ checkedOutStatus = dom\ checkedOutBy = dom\ permission = \\
&dom\ category = dom\ documentTitle = documentId \\
&dom\ accessLevel = ran\ checkedOutBy = ran\ permission = userId
\end{aligned}
$$

Question 2 (Continued)

a) Design a schema called *CheckOut* that allows a user to check out a document for editing. (10 marks)

b) Design a schema called *CheckIn* that allows a user to check in a document after editing, making it available for others to edit. A a new version of the document is created. (11 marks)

c) Design a schema called *DisplayUsers* that shows all users with read and write access level. (4 marks)

[Total: 25 marks]

Question 3

Consider a scenario where you are tasked with modeling a queuing system at an airport check-in counter. In this system, all passengers must first join a single queue before they can check in for their flights. Although there are multiple check-in counters available, passengers are served in the order they reach the front of the line. Once at the front, they are then directed to the next available counter for check-in.

The system follows a first-come, first-served basis. This means that passengers are attended to in the exact order they arrive, and no passenger can be served before the person ahead of them in the queue. While waiting, the passengers in queue are not the same individuals who are being assisted at the counters.

If a counter is empty, it indicates that there are no passengers currently waiting for check-in service, and the system may have no one queued at that moment.

This system is designed to ensure that check-in is fair and efficient, with passengers being processed in a structured order. The number of open counters, the time each passenger takes to check in, and the arrival rate of passengers can all affect how quickly the check-in process moves forward.

You are given with two basic types

[COUNTER] - the set of all check-in counters in the airport
[PASSENGER] - the set of all airport passengers

a) Based on the information provided, summarise the information into a state space schema called *PassengerCheckIn* that includes the following components:

- *counters* which represents all the available check-in counters for serving passengers.
- *queue* which represents all the passengers waiting in line to be served at the check-in counters.
- *checkIn* which represents the relationship between the check-in counters and the passengers being served at those counters.

You must specify all possible invariants (conditions) in the predicate part of your state space schema. (6 marks)

Question 3 (Continued)

b) Interpret the specification of *ServePassenger* stated below and illustrate it using Z schema.

Use case:	ServePassenger
Purpose:	To serve the passenger at the front of the queue for check-in.
Pre-conditions:	1) The counter must exist in the system and must be empty for check-in. 2) The queue must not be empty. 3) The passenger is queuing and must be at the front of the line.
Initiating actor:	Check-in Agent
Normal flow:	1) The passenger is input into the system. 2) The counter is input into the system. 3) System confirms the counter exists in the system and empty for check-in. 4) System confirms the queue is not be empty. 5) System confirms the passenger is queuing and at the front of the line. 6) System updates the information 7) Exit success
Exceptions flow:	a) Counter does not exist in the system a1 Exit failure b) Counter is not empty b1 Exit failure c) Queue is not empty c1 Exit failure d) Passenger is not queuing d1 Exit failure e) Passenger is not at the front of the line e1 Exit failure

(8 marks)

c) Design a schema called *QueueLineUp* that allows a passenger to join the end of the queue for check-in. (5 marks)

d) The schema *QueueLineUp* from **Question 3 c)** must be refined to deal with error exceptions. Given the messages for the error exception in a free type definition as below:

RESPONSE ::= okay | counterExist | counterNotExist | counterFree | counterFull
| queueEmpty | queueNotEmpty | atFrontLine | notAtFrontLine
| passengerAlreadyQueuing | passengerNotQueuing
| passengerAtCounter | passengerNotAtCounter

(i) Develop an error exception schema that will provide an error message for every error in the schema *QueueLineUp*. (4 marks)

(ii) Assume that you already produce a successful schema called *Success*. Prepare a complete schema called *QueueLineUpComplete* using schema definition notation which caters for all possible violations of the precondition in the schema *QueueLineUp* from **Question 3 c)**. (2 marks)

[Total: 25 marks]

Question 4

a) Differentiate between *schema inclusion* and *schema hiding* in the Z language. (8 marks)

b) *Bag* is one of the structures used in Z schema to represent data.

(i) Using *bag* notation to model a real-life example, provide a detailed explanation of the concept of a *bag*. (5 marks)

(ii) Common operators in a *bag* include *items* and *count*. Using appropriate examples, point out how *items* and *count* are used in *bag*. (6 marks)

c) Examine the following expressions and produce your results for each expression by showing the steps:

(i) $\text{front}\,((\langle q, i, c, k\rangle \oplus \{2 \mapsto u\}) \frown \langle t, h, i, n, k, i, n, g\rangle) \upharpoonright \langle i, o, k, g\rangle$ (3 marks)

(ii) $((\langle i, m, p, r, o, v, e\rangle \frown \langle s, t, u, d, y\rangle) \oplus \{5 \mapsto v\}) \sim (\!|\{r, t, v\}|\!)$ (3 marks)

[Total: 25 marks]