Victoria University of Wellington

DEGREE EXAMINATIONS — 1999

MID-YEAR

COMP 426

FORMAL SOFTWARE DEVELOPMENT

Time Allowed: 3 Hours

Instructions: There are 4 questions, worth 25 marks each.
Answer all questions.
Make sure your answers are clear, complete and to the point.

No reference material is allowed.
No calculators are allowed.
Foreign language dictionaries *are* allowed.

A sheet summarizing the Z notation and the logical rules is provided.

Question 1. [25 marks]

(a) [2 marks] We may define a set using a *set comprehension*, using the following syntax:

$$\{x : T \mid P \bullet E\}$$

Explain the roles of T, P and E in this expression.

(b) [4 marks] Use set comprehensions to define the following:

(i) $\operatorname{dom} R$, the domain of a relation R
(ii) $\operatorname{ran} R$, the range of a relation R
(iii) $A \lhd R$, the domain restriction of R to A
(iv) $R \rhd A$, the range restriction of R to A

(c) [3 marks] A schema can be written as:

```
┌─ W[X] ──────────────────────────
│ Y
├──────────
│ Z
└─────────────────────────────────
```

Explain the roles of W, X, Y and Z.

(Question 1 continued on next page)

(Question 1 continued)

(d) [8 marks] Explain the following operations on schemas:
(i) inclusion
(ii) decoration
(iii) disjunction
(iv) conjunction

(e) [8 marks] An airline has been having problems with the computer system it uses to sell flight tickets and to allocate seats on its aeroplanes, and has employed you to help.

There are three types of ticket:

- Prestige
- Economy
- Stand-by

There are two classes of seating on the plane:

- First-class
- Second-class

The airline will sell up to ten more tickets for a flight than there are seats on the plane. They will only sell as many Prestige tickets as there are First-class seats.

A holder of Stand-by ticket will be allocated a Second-class seat if there is one free. If there is not the passenger won't get to fly.

A holder of an Economy ticket will be allocated a Second-class seat if there is one free. If there is not, and there is a Second-class seat held by a Stand-by passenger, then the Stand-by passenger will lose the seat to the Economy passenger.

The holder of a Prestige ticket will be allocated a First-class seat.

Present Z declarations and schemas to specify:
(i) the state of the system
(ii) selling tickets
(iii) allocating seats to passengers

Question 2. [25 marks]

(a) [6 marks] For each of the following *either* prove it, *or* explain why it cannot be proved:

(i) $\vdash A \Rightarrow B \Rightarrow A$

(ii) $B \vee A \vdash \neg (\neg A \wedge \neg B)$

(iii) $B \Rightarrow C, A \Rightarrow B \vdash A \Rightarrow C$

(b) [3 marks] The connective $\Leftrightarrow$ may be defined as follows:

$$A \Leftrightarrow B == (A \Rightarrow B) \wedge (B \Rightarrow A)$$

Present suitable introduction and elimination rules for $\Leftrightarrow$.

(c) [4 marks] *Either* prove the following, *or* explain why it cannot be proved:

$$\vdash A \Leftrightarrow \neg \neg A$$

(d) [12 marks] Formal methods are not as widely used in the computing industry as academic advocates of formal methods believe they should be. Explain why this is the case. You may want to discuss the following:

- the place of formal methods in the software lifecycle
- weaknesses (real and perceived) of formal methods
- advantages (real and perceived) of formal methods
- case studies

Question 3. [25 marks]

Consider the following specification, which describes a system for managing a class database. The database contains information about:

- which students are enrolled for this class
- whether students have completed the work for the course
- the final grade (A, B, C, or F) for each student.

Of course, a student cannot get a grade without completing the work, and cannot complete the work without being enrolled.

$[STUDENT]$
$GRADE ::= A \mid B \mid C \mid F$

$$
\begin{array}{|l}
\hline
\textit{Class} \\
\textit{enrolled} : \mathbb{F}\, STUDENT \\
\textit{completed} : \mathbb{F}\, STUDENT \\
\textit{grade} : STUDENT \pfun GRADE \\
\hline
\textit{completed} \subseteq \textit{enrolled} \\
\operatorname{dom} \textit{grade} \subseteq \textit{completed} \\
\hline
\end{array}
$$

Initially, the class has no students:

$$
\begin{array}{|l}
\hline
\textit{InitClass} \\
\textit{Class}' \\
\hline
\textit{enrolled}' = \varnothing \\
\hline
\end{array}
$$

There are operations to enrol a student, and to record that a student has completed the work:

$$
\begin{array}{|l}
\hline
\textit{Enrol} \\
\Delta \textit{Class} \\
s? : STUDENT \\
\hline
s? \notin \textit{enrolled} \\
\textit{enrolled}' = \textit{enrolled} \cup \{s?\} \\
\textit{completed}' = \textit{completed} \\
\hline
\end{array}
$$

(Question 3 continued on next page)

(Question 3 continued)

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Complete}} \\
\Delta Class \\
s? : STUDENT \\
\hline
enrolled' = enrolled \\
completed' = completed \cup \{s?\} \\
grade' = grade \\
\hline
\end{array}
$$

Finally, we can mark the completed work:

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Mark}} \\
\Delta Class \\
s? : STUDENT \\
g? : GRADE \\
\hline
enrolled' = enrolled \\
completed' = completed \\
grade' = grade \cup \{s? \mapsto g?\} \\
\hline
\end{array}
$$

(a) [3 marks] Show that the initialization produces a valid initial state of the *Class* system.

(b) [3 marks] What can we conclude about the values of $enrolled'$, $completed'$ and $grade'$ immediately after initialization?

(c) [3 marks] Calculate the precondition for the *Complete* operation, and explain the result of the calculation in words.

(d) [4 marks] Suppose the class administrator has entered a grade of B for student Joe:

$$Mark[Joe/s?, B/g?]$$

but Joe convinces the marker to change his grade to A. The class administrator invokes the *Mark* operation again, expecting to update the recorded grade:

$$Mark[Joe/s?, A/g?]$$

Explain why this is unsatisfactory, and give two different ways of modifying the *Mark* schema to avoid this situation.

(Question 3 continued on next page)

(Question 3 continued)

The class administrator now needs to update the University's central database, which records the status of students in the class as follows:

$$STATUS ::= A \mid B \mid C \mid F \mid N$$

$$\begin{array}{|l}
\hline
\textit{ClassRegistry} \\
status : STUDENT \rightarrowtail\!\!\!\!\!\!+ \; STATUS \\
\hline
\end{array}$$

The *ClassRegistry* records a status of A, B, C, or F for each student who has completed the course and obtained the specified grade. It records a status of N (for "not complete") for those students who have enrolled for, but not yet completed, the course.

(e) [3 marks] Complete the definition below for the schema *Rel*, representing a suitable abstraction relation for data refining *Class* to *ClassRegistry*.

$$\begin{array}{|l}
\hline
\textit{Rel} \\
\textit{Class} \\
\textit{ClassRegistry} \\
\hline
\ldots \\
\hline
\end{array}$$

(f) [4 marks] Initially, the *ClassRegistry* has no information about students:

$$\begin{array}{|l}
\hline
\textit{InitClassRegistry} \\
\textit{ClassRegistry}' \\
\hline
status' = \varnothing \\
\hline
\end{array}$$

Using your answer to part (e) above, show that this initialization is a data refinement of the corresponding initialization of the *Class* system.

(g) [5 marks] Using your answer to part (e) above, calculate a refinement for the *Mark* operation.

Question 4. [25 marks]

(a) [3 marks] A command S is *conjunctive* if, for any predicates ϕ and ψ, we have:

$$wp(S, \phi \wedge \psi) \equiv wp(S, \phi) \wedge wp(S, \psi)$$

Prove that every specification statement $w\!: \left[P \;\middle/\; Q \right]$ is conjunctive.

(b) [4 marks] A command S is *disjunctive* if, for any predicates ϕ and ψ, we have:

$$wp(S, \phi \vee \psi) \equiv wp(S, \phi) \vee wp(S, \psi)$$

Exhibit a command that is not disjunctive, and give ϕ and ψ to prove this.

(c) [3 marks] The refinement law for introducing a loop may be written as follows:

$$\frac{P \Rightarrow I \qquad\qquad I \wedge \neg B \Rightarrow Q}{w\!: \left[P \;\middle/\; Q \right] \sqsubseteq \textbf{do } B \rightarrow w\!: \left[I \wedge B \;\middle/\; I \wedge 0 \leq t < t_0 \right] \textbf{od}}$$

Explain this law.

(d) [15 marks] Consider the following requirements:

> Given a non-empty array A of n integers, compute:
>
> - m, the value of the minimum element; and
> - c, the number of times m occurs.
>
> Do *not* assume there is a min operator for finding the minimum.

The following specification statement expresses these requirements:

$$m, c\!: \left[n \geq 1 \;\middle/\; \begin{array}{c} \forall\, i \mid 1 \leq i \leq n \bullet A[i] \geq m \\ \exists\, i \mid 1 \leq i \leq n \bullet A[i] = m \\ c = \#\{ i \mid 1 \leq i \leq n \bullet A[i] = m \} \end{array} \right]$$

(i) Present a suitable invariant, I, for a loop satisfying this specification. Recall that a loop invariant should: be easy to establish initially; be able to be extended one step at a time; guarantee the postcondition on termination.

(ii) Give an appropriate initialization for the loop. State what must be proved to show that the initialization is valid, and briefly justify why these proof obligations hold.

(iii) Use the refinement law from part (c) above, with your invariant I from part (i), to introduce the loop. Give the loop structure, and state and justify each proof obligation. Leave a specification statement as the body of your loop.

(iv) Summarize the remainder of the refinement to code. You do not need to give a full proof, but should indicate what refinement laws are needed and what the major proof obligations will be.

****