VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wānanga o te Ūpoko o te Ika a Māui

EXAMINATIONS — 2004

MID-YEAR

COMP 426

Formal Software Development

Time Allowed: 3 Hours

Instructions: Candidates should attempt **all FOUR** questions.

This exam will be marked out of 100.

Foreign language translation dictionaries are allowed.

A summary of Z mathematical notation is provided.

Question 1. Writing formal specifications in Z [30 marks]

You are asked to write a formal specification for a simple appointments diary system, using Z. Each appointment has a start time, an end time, and a description. We will treat times as natural numbers, so that times are ordered and we can add and subtract times. We are otherwise not concerned with what time values actually represent, so we do not specify what time units they correspond to. The time period for an appointment is modelled as a pair of times, giving its start time and end time; for example, (t_1, t_2) represents the time period starting at time t_1 and ending at time t_2, and includes both t_1 and t_2. Time t is within period (t_1, t_2) if $t_1 \leq t \leq t_2$, and two periods overlap if some time is within both periods. Appointment descriptions are assumed to be of type *Text*, which is not further specified.

$$
\begin{aligned}
&\mathit{Time} == \mathbb{N} \\
&\mathit{Period} == \mathit{Time} \times \mathit{Time} \\
&[\mathit{Text}]
\end{aligned}
$$

The state of the diary system is modelled as a set of appointment periods and a mapping from these appointments to their descriptions.

$$
\begin{array}{|l}
\hline
\textit{Diary} \\
\mathit{appointments} : \mathbb{F}\ \mathit{Period} \\
\mathit{description} : \mathit{Period} \mathrel{+\!\!\!\!\twoheadrightarrow} \mathit{Text} \\
\hline
\mathrm{dom}\ \mathit{description} = \mathit{appointments} \\
\hline
\end{array}
$$

(a) [2 marks]

Write a Z schema describing the initial state of the diary system, in which there are no appointments recorded.

Show that the initial state described by your schema satisfies the state invariant of the *Diary* system.

(b) [4 marks]

Complete the following specification for a *Schedule* operation, which adds an appointment to the diary, provided that the time period for new appointment does not overlap with the time period for any existing appointment. Note that two time periods overlap if some time is

$$
\begin{array}{|l}
\hline
\textit{Schedule} \\
\Delta \mathit{Diary} \\
\mathit{start}? : \mathit{Time} \\
\mathit{end}? : \mathit{Time} \\
\mathit{desc}? : \mathit{Text} \\
\hline
\ldots \\
\hline
\end{array}
$$

(c) [4 marks]

Complete the following specification for a *Cancel* operation, which removes the appointment with a given start time from the diary, provided that there is an appointment starting at that time.

$$
\begin{array}{|l}
\hline
\textit{Cancel} \\
\Delta \textit{Diary} \\
\textit{start}? : \textit{Time} \\
\hline
\ldots \\
\hline
\end{array}
$$

(d) [4 marks]

Write a specification for a *Details* operation, which returns the description of the appointment with a given start time, provided that there is an appointment starting at that time.

(e) [4 marks]

Explain what it means for a specification to be *robust*. Show how operations *Schedule*, *Cancel* and *Details* can be extended to obtain a robust version of the diary system, explaining how the robust operations are constructed.

(f) [12 marks]

Suppose we now want to extend the diary system so that it maintains appointment diaries for a group of people.

(i) Write a state schema for the extended system.

(ii) Explain how the operations in the *Diary* system can be extended to obtain corresponding operations for the extended system, using *promotion*.

You do not need to give a complete specification for the resulting system, just enough to illustrate the use of promotion.

(iii) Define an operation to schedule a meeting for several people. The operation should take as inputs the set of people who need to attend the meeting, and the length and description of the meeting. It should, if possible, find a time at which all of the people can attend the meeting, add it to their diaries, and return its start time as output; if there is no time at which all of the people can attend a meeting of the required duration, the operation should return 0 as its output and leave the diaries unchanged. For the purposes of this question, we will assume that no appointment starts at time 0.

You will get additional marks if your operation finds the *earliest* time at which the meeting can be scheduled.

Question 2. Data Refinement in Z [30 marks]

Below is a specification for a system to keep track of the membership of parliamentary committees. The system records a set of MPs; a set of committees; and for each committee, its chairperson and its other members.

$[MP, Committee]$

$$
\begin{array}{|l}
\hline
\textit{Committees} \\
\hline
mps : \mathbb{F}\, MP \\
committees : \mathbb{F}\, Committee \\
chair : Committee \pfun MP \\
members : Committee \pfun \mathbb{F}\, MP \\
\hline
\operatorname{dom} chair = committees \\
\operatorname{ran} chair \subseteq mps \\
\operatorname{dom} members = committees \\
\forall c : committees \bullet \\
\quad members(c) \subseteq mps \setminus \{chair(c)\} \\
\hline
\end{array}
$$

The system is initialised with a set of MPs; initially there are no committees.

$$
\begin{array}{|l}
\hline
\textit{InitCommittees} \\
\hline
Committees' \\
mps? : \mathbb{F}\, MP \\
\hline
mps' = mps? \\
committees' = \varnothing \\
\hline
\end{array}
$$

There is an operation to specify the chairperson and members of a committee. This operation can be used to create a new committee, or to replace the details for an existing committee.

$$
\begin{array}{|l}
\hline
\textit{SpecifyMembers} \\
\hline
\Delta Committees \\
cmte? : Committee \\
chair? : MP \\
members? : \mathbb{F}\, MP \\
\hline
chair' = chair \oplus \{cmte? \mapsto chair?\} \\
members' = members \oplus \{cmte? \mapsto members?\} \\
\hline
\end{array}
$$

There is an operation to list the chair and members of a given committee. This returns a sequence of MPs on the committee, with the chairperson listed first.

$$
\begin{array}{|l}
\hline
\textit{ListMembers} \\
\Xi \textit{Committees} \\
\textit{cmte}? : \textit{Committee} \\
\textit{members}! : \mathrm{iseq}\, \textit{MP} \\
\hline
\textit{cmte}? \in \textit{committees} \\
\textit{head}\ \textit{members}! = \textit{chair}(\textit{cmte}?) \\
\mathrm{ran}(\textit{tail}\ \textit{members}!) = \textit{members}(\textit{cmte}?) \\
\hline
\end{array}
$$

There is an operation to list the committees a given MP is on. This returns a list of the committees for which the MP is chairperson, and a list of committees for which the MP is a member but not chairperson.

$$
\begin{array}{|l}
\hline
\textit{ListCommittees} \\
\Xi \textit{Committees} \\
\textit{mp}? : \textit{MP} \\
\textit{chair}! : \mathrm{iseq}\, \textit{Committee} \\
\textit{member}! : \mathrm{iseq}\, \textit{Committee} \\
\hline
\textit{mp}? \in \textit{mps} \\
\mathrm{ran}\, \textit{chair}! = \textit{chair}^{\sim} (\!| \{\textit{mp}?\} |\!) \\
\mathrm{ran}\, \textit{member}! = \{ c : \textit{committees} \mid \textit{mp}? \in \textit{members}(c) \} \\
\hline
\end{array}
$$

We data refine the system using a representation which records a list of MPs ($mps1$), a list of committees ($committees1$), and a list ($members1$) in which the ith element is a list of the members of the ith committee in the committee list, with the chairperson listed first.

$$
\begin{array}{|l}
\hline
\textit{Committees}1 \\
\textit{mps}1 : \mathrm{iseq}\, \textit{MP} \\
\textit{committees}1 : \mathrm{iseq}\, \textit{Committee} \\
\textit{members}1 : \mathrm{iseq}(\mathrm{iseq}\, \textit{MP}) \\
\hline
\# \textit{committees}1 = \# \textit{members}1 \\
\forall\, i : \mathrm{dom}\, \textit{members}1 \bullet \mathrm{ran}(\textit{members}1(i)) \subseteq \mathrm{ran}\, \textit{mps}1 \\
\hline
\end{array}
$$

(a) [6 marks]

Write a schema defining an abstraction relation between *Committees* and *Committees*1. Explain how your abstraction relation captures the relationship between *Committees* and *Committees*1.

(b) [20 marks]

Write Z schemas describing the initial state of *Committees*1, and "concrete" operations *SpecifyMembers*1, *ListMembers*1 and *ListCommittees*1, which data refine the initialisation and operations of *Committees*, using the above representation.

Explain briefly why each schema satisfies the proof obligations for data refinement.

(c) [4 marks]

Explain how an abstraction relation can be used to *calculate* a minimally refined data refinement for a Z specification. In what ways would you expect the calculated data refinement for the *Committees* system to differ from the one you gave in part **(b)**?

Question 3. Weakest precondition semantics [18 marks]

(a) [6 marks] Define the following properties of a statement S, in terms of its weakest precondition semantics:

(i) monotonic (with respect to implication)

(ii) feasible (or strict)

(iii) terminating

(iv) disjunctive

(v) conjunctive

(vi) continuous

(b) [2 marks] Which of the above properties does Dijkstra require as *healthiness conditions*?

(c) [4 marks] Which of Dijkstra's healthiness conditions are **not** required in the refinement calculus? In each such case, explain why that property is not appropriate in a wide-spectrum language, and give an example of a construct in the refinement calculus which does not have that property.

(d) [3 marks]

Define the weakest precondition for a specification statement, and show that specification statements are conjunctive.

(e) [3 marks]

Define the weakest precondition for sequential composition, and show that sequential composition preserves disjunctivity (i.e. show that if S and T are disjunctive, then so is $S \,;\, T$).

(You may assume that sequential composition is monotonic with respect to refinement.)

Question 4. Refinement calculus [22 marks]

The following is an incomplete derivation of a program in the refinement calculus, where $x \in A[1..n]$ is an abbreviation for $\exists\, i : 1..n \bullet x = A[i]$.

$$p : \left[\, x \in A[1..n] \;\middle/\; p \in 1..n \wedge A[p] = x \,\right] \qquad (1)$$

(1) $\sqsubseteq$ ⟨ *introduce variable* ⟩

$\llbracket$ **var** q $\bullet$

$p, q : \left[\, x \in A[1..n] \;\middle/\; p \in 1..n \wedge A[p] = x \,\right]$ (2)

$\rrbracket$ Ob: k *fresh*

(2) $\sqsubseteq$ ⟨ *split specification* ⟩

$p, q : \left[\, x \in A[1..n] \;\middle/\; Inv \,\right]$; (3)

$p, q : \left[\, Inv \;\middle/\; p \in 1..n \wedge A[p] = x \,\right]$ (4)

where $Inv \mathrel{\widehat{=}} 1 \le p \le q \le n \wedge x \in A[p..q]$

(3) $\sqsubseteq$ ⟨ *introduce assignment* ⟩

$p, q := 1, n$

Ob: $x \in A[1..n] \Rightarrow Inv[p, q \setminus 1, n]$

(4) $\sqsubseteq$ ⟨ *introduce* **do** ⟩

do $p \neq q \rightarrow$

$p, q : \left[\, Inv \wedge p \neq q \;\middle/\; Inv \wedge 0 \le q - p < q_0 - p_0 \,\right]$ (5)

od

Obs: $Inv \Rightarrow Inv$

$Inv \wedge p = q \Rightarrow p \in 1..n \wedge A[p] = x$

(5) $\sqsubseteq$ ⟨ *introduce* **if** ⟩

if $A[p] = x \rightarrow$

$p, q : \left[\, Inv \wedge p \neq q \wedge A[p] = x \;\middle/\; Inv \wedge 0 \le q - p < q_0 - p_0 \,\right]$ (6)

$\talloblong$ $A[q] = x \rightarrow$

$p, q : \left[\, Inv \wedge p \neq q \wedge A[q] = x \;\middle/\; Inv \wedge 0 \le q - p < q_0 - p_0 \,\right]$ (7)

$\talloblong$ $A[p] \neq x \wedge A[q] \neq x \rightarrow$

$p, q : \left[\, Inv \wedge p \neq q \wedge A[p] \neq x \wedge A[q] \neq x \;\middle/\; Inv \wedge 0 \le q - p < q_0 - p_0 \,\right]$ (8)

fi

Ob: $Inv \wedge p = q \Rightarrow A[p] = x \vee A[q] = x \vee A[p] \neq x \wedge A[q] \neq x$

(6) $\sqsubseteq$ $\langle$ *introduce assignment* $\rangle$

$q := p$

Ob: $Inv \wedge p \neq q \wedge A[p] = x \Rightarrow (Inv \wedge 0 \leq q{-}p < q_0{-}p_0)[p \setminus q]$

(7) $\sqsubseteq$ $\langle$ *introduce* **skip** $\rangle$

$p := q$

Ob: $Inv \wedge p \neq q \wedge A[q] = x \Rightarrow (Inv \wedge 0 \leq q{-}p < q_0{-}p_0)[q \setminus p]$

(8) $\sqsubseteq$ $\langle$ *introduce* **skip** $\rangle$

$p, q := p + 1, q - 1$

Ob: $Inv \wedge p \neq q \wedge A[p] \neq x \wedge A[q] \neq x \Rightarrow$
$(Inv \wedge 0 \leq q{-}p < q_0{-}p_0)[p, q \setminus p + 1, q + 1]$

(a) [3 marks] Explain what the specification statement labelled (1) means, identifying each of the components of the specification statement and explaining how it contributes to the meaning of the whole statement.

(b) [7 marks] For each step in the above derivation, give the *proof obligation(s)* that must be satisfied, and explain briefly why they are valid.

(c) [3 marks] What is the *loop invariant* used in the above derivation? Explain the rôle of the loop invariant in ensuring the correctness of the program.

(d) [3 marks] What is the *variant* or *bound* function used in the above derivation? Explain how it is used to ensure that the loop terminates.

(e) [6 marks] Show how the specification statements labelled (6) and (8) can be refined to code.

Indicate the laws used, write down the necessary proof obligations and briefly explain why they are valid.

****