# DEGREE EXAMINATIONS — 1995



(END OF YEAR)

COMP 426

Formal Software Development

Time Allowed: THREE hours.

Instructions: Candidates should attempt all questions.
This exam will be marked out of 100.

**Question 1.** **[30 marks]**

**(a)** In Diller's **Z** specification of the Library Problem, the state of the library system is defined using three schemas, as follows:

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\_ \mathit{ParaLibrary} \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\mathit{instanceof} : \mathit{Copy} \pfun \mathit{Book} \\
\mathit{writtenby} : \mathit{Book} \pfun \mathbb{F}\, \mathit{Author} \\
\mathit{about} : \mathit{Book} \pfun \mathbb{F}\, \mathit{Subject} \\
\hline
\mathrm{dom}\, \mathit{writtenby} \subseteq \mathrm{ran}\, \mathit{instanceof} \\
\mathrm{dom}\, \mathit{about} \subseteq \mathrm{ran}\, \mathit{instanceof} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{LibraryDB}} \\
\hline
borrower, staff : \mathbb{F}\, Person \\
available, checkedout : \mathbb{F}\, Copy \\
previouslyborrowedby, borrowedby : Copy \twoheadrightarrow Person \\
\hline
borrower \cap staff = \varnothing \\
available \cap checkedout = \varnothing \\
\mathrm{dom}\, borrowedby = checkedout \\
\mathrm{ran}\, borrowedby \subseteq borrower \\
\mathrm{dom}\, previouslyborrowedby \subseteq available \cup checkedout \\
\mathrm{ran}\, previouslyborrowedby \subseteq borrower \\
\forall\, p : Person \mid p \in borrower \bullet \\
\qquad \#(borrowedby^{\sim}(\!|\{p\}|\!)) \leq MaxCopiesAllowed \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{LibraryState}} \\
\hline
ParaLibrary \\
LibraryDB \\
\hline
\mathrm{dom}\, instanceof = available \cup checkedout \\
\hline
\end{array}
$$

Explain why the specification is structured in this way. What advantage does this structure have over using a single schema to define the state? Would a different structure be better than this one? [4 marks]

**(b)** The basic operations for checking out and returning a book (ignoring error conditions) are defined using the following schemas:

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{CheckOutCopy}} \\
\hline
\Delta LibraryState \\
\Xi ParaLibrary \\
n? : Person \\
c? : Copy \\
\hline
n? \in borrower \\
c? \in available \\
\#(borrowedby^{\sim}(\!|\{n?\}|\!)) < MaxCopiesAllowed \\
available' = available \setminus \{c?\} \\
checkedout' = checkedout \cup \{c?\} \\
borrowedby' = borrowedby \cup \{c? \mapsto n?\} \\
previouslyborrowedby' = previouslyborrowedby \\
borrower' = borrower \\
staff' = staff \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{Return} \\
\Delta \mathit{LibraryState} \\
\Xi \mathit{ParaLibrary} \\
c? : \mathit{Copy} \\
\hline
c? \in \mathit{checkedout} \\
\mathit{available}' = \mathit{available} \cup \{c?\} \\
\mathit{checkedout}' = \mathit{checkedout} \setminus \{c?\} \\
\mathit{borrowedby}' = \{c?\} \mathbin{\lhd} \mathit{borrowedby} \\
\mathit{previouslyborrowedby}' = \\
\quad \mathit{previouslyborrowedby} \oplus \{c? \mapsto \mathit{borrowedby}(c?)\} \\
\mathit{borrower}' = \mathit{borrower} \\
\mathit{staff}' = \mathit{staff} \\
\hline
\end{array}
$$

(i) Show that the invariants

- $\operatorname{dom} \mathit{previouslyborrowedby} \subseteq \mathit{available} \cup \mathit{checkedout}$, and
- $\operatorname{ran} \mathit{previouslyborrowedby} \subseteq \mathit{borrower}$

are preserved by these operations. [4 marks]

(ii) Derive, from these specifications, the net effect of a user checking out a book and then immediately returning it. [4 marks]

**(c)** Suppose the library management decides that instead of just recording the previous borrower of each copy, the system should keep a complete borrowing history for each copy detailing each user who borrows a book, the date on which they borrowed it, and the date on which they returned it.

Show the changes required to the above specification in order to accommodate this requirement. Explain your changes. [10 marks]

**(d)** Suppose you are asked to implement part of the library system. The project leader has decided to record information about copies currently on loan by keeping a file with a record for each borrower containing an array with details of all the copies they currently have checked out.

Write a Z schema defining this data structure, including the representation invariant.

Write a retrieve relation (as a schema) showing the relationship between this data structure and *borrowed* in the original state. [8 marks]

**Question 2.** **[18 marks]**

**(a)** Define the weakest precondition semantics for **if** $[]_i\ G_i \rightarrow S_i$ **fi**, as in Dijkstra's guarded commands language. [2 marks]

**(b)** Define the weakest precondition semantics for the constructs $G \rightarrow S$, $S_1 [] S_2$ and **if** $S$ **fi**, as separate commands.

Compare the resulting semantics for **if** $G_1 \rightarrow S_1 [] G_2 \rightarrow S_2$ **fi** with Dijkstra's definition. [10 marks]

**(c)** State Dijkstra's monotonicity condition for weakest preconditions. Prove that the weakest preconditions for $G \rightarrow S$, $S_1 [] S_2$ and **if** $S$ **fi** satisfy this condition, assuming that $S$, $S_1$ and $S_2$ do. [6 marks]

**Question 3.** **[14 marks]**

**(a)** Show that **if** $[]_i\ G_i \rightarrow S_i$ **fi** (as defined by Dijkstra) is monotonic with respect to refinement. [3 marks]

**(b)** Show (by giving a counter-example) that **if** $S$ **fi** is not monotonic with respect to refinement. [3 marks]

**(c)** Prove that the following refinement rule is valid:

$$\textit{IF1:} \quad \frac{P \Rightarrow \bigvee_i B_i}{\tilde{w}:[\,P\,,\ Q\,] \ \sqsubseteq\ \mathbf{if}\ \ []_i\ B_i \rightarrow \tilde{w}:[\,P \wedge B_i\,,\ Q\,]\ \mathbf{fi}}$$

[4 marks]

**(d)** The following is an alternative rule for introducing **if** commands:

$$\textit{IF2:} \quad \frac{\bigvee_i B_i}{\tilde{w}:[\,P\,,\ Q\,] \ \sqsubseteq\ \mathbf{if}\ \ []_i\ B_i \rightarrow \tilde{w}:[\,P \wedge B_i\,,\ Q\,]\ \mathbf{fi}}$$

Show that *IF2* is weaker than *IF1*, by giving a refinement which can be proved using *IF1* but not using *IF2*. [2 marks]

**(e)** Show (by giving a counter-example) that *IF2* would not be valid if the applicability condition $\bigvee_i B_i$ was omitted. [2 marks]

**Question 4.** **[22 marks]**

The following specification statement specifies a program to determine whether the elements of a sequence, $A$, are in (strictly) ascending order:

$$z\!:\![\, true\, ,\ z \equiv asc(A)\,] \tag{1}$$

where: $\#A$ is the length of $A$, and

$$asc(A) \;\widehat{=}\; (\forall\, i, j : 1..\#A \bullet i < j \Rightarrow A[i] < A[j])$$

**(a)** Suppose you decide to implement this specification by counting the number of ascending adjacent pairs and then comparing this count with $\#A - 1$. Show that the following is a valid refinement of specification statement (1):

$$\begin{array}{l} |[\ \textbf{var}\ c \bullet \\ \quad c\!:\![\, true\, ,\ c = \#\{i : 1\,..\,\#A - 1 \mid A[i] < A[i+1]\}\,]; \\ \quad z := (c = \#A - 1) \\ ]| \end{array} \tag{2}$$

Indicate the refinement rule used at each step in the derivation, and give a brief justification of any applicability conditions. [4 marks]

**(b)** Explain how a *loop invariant* can be obtained from specification statement (2), using Gries' strategy of *weakening the postcondition*, and show the resulting loop invariant. [4 marks]

**(c)** Derive an algorithm that satisfies specification statement (2), using the refinement calculus. Indicate the refinement rule used at each step in the derivation, and give a brief justification of any applicability conditions. [12 marks]

**(d)** The above implementation of (1) is inefficient, because it needs to inspect every pair of adjacent elements in $A$. A better implementation would stop as soon as a non-ascending adjacent pair was found. Give a loop invariant which would lead to this implementation. [2 marks]

**Question 5.** **[16 marks]**

The "data refinement" relation ($\preceq$) on programs can be defined as follows:

$S \preceq_{rep} S'$ iff $rep \circ S \;\sqsubseteq\; S' \circ rep$
for some predicate transformer *rep* which is monotonic and $\vee$-distributive.

**(a)** Prove that data refinement distributes through sequential composition, i.e. show that if $S_1 \preceq_{rep} S_1'$ and $S_2 \preceq_{rep} S_2'$, then $S_1; S_2 \preceq_{rep} S_1'; S_2'$. [4 marks]

**(b)** Prove that data refinement is semi-transitive, i.e. show that if $S_1 \preceq_{rep_1} S_2$ and $S_2 \preceq_{rep_2} S_3$, then $S_1 \preceq_{rep_3} S_3$, for some $rep_3$. [4 marks]

**(c)** Suppose that *rep* is defined as $rep\,\phi \;\widehat{=}\; (\exists\, a \bullet I \wedge \phi)$, where $I$ is a relation between the abstract variables $a$ and concrete variables $c$. Show that the above definition reduces to the alternative definition of data refinement:

$S \preceq_{a,I,c} S'$ iff $(\exists\, a \bullet I \wedge wp(S, \phi)) \Rightarrow wp(S', (\exists\, a \bullet I \wedge \phi))$
for any assertion $\phi$ not containing any variables in $c$.

[4 marks]

**(d)** Explain what is meant by *adequacy* of a data representation. Discuss, with examples, whether adequacy is a necessary requirement for data refinement. [4 marks]

*******************************