Victoria University of Wellington

DEGREE EXAMINATIONS — 1997

END OF YEAR

COMP 426

Formal Software Development

Time Allowed: 3 Hours

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

This exam will be marked out of 100.

Question 1. **[18 marks]**

Suppose you wish to describe the rules of chess using Z.

(a) Explain how you would model the current state of a chess game in Z, and give the Z declarations required. [6 marks]

(b) Give a specification in Z describing the legal moves (including capture moves) for one variety of chess piece (i.e. for king, queen, bishop, knight, rook or pawn).

You should explain the move being specified in English, and explain how this is captured by the formal specification. [6 marks]

(c) Discuss the features provided in Z for structuring formal specifications, with reference to structuring the specification of the rules of chess. In particular, discuss the way in which you would specify the condition for a winning position by combining specifications of other aspects of the game. [6 marks]

Question 2. **[14 marks]**

The following schema definitions are taken from Spivey's specification and data refinement of the birthday book system:

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{BirthdayBook}} \\
\hline
\mathit{known} : \mathbb{P}\, \mathit{NAME} \\
\mathit{birthday} : \mathit{NAME} \pfun \mathit{DATE} \\
\hline
\mathit{known} = \operatorname{dom} \mathit{birthday} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{BirthdayBook1}} \\
\hline
\mathit{names} : \mathbb{N}_1 \to \mathit{NAME} \\
\mathit{dates} : \mathbb{N}_1 \to \mathit{DATE} \\
\mathit{hwm} : \mathbb{N} \\
\hline
\forall\, i, j : 1 \mathinner{..} \mathit{hwm} \bullet \\
\qquad i \neq j \Rightarrow \mathit{names}(i) \neq \mathit{names}(j) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{AddBirthday}} \\
\hline
\Delta \mathit{BirthdayBook} \\
\mathit{name?} : \mathit{NAME} \\
\mathit{date?} : \mathit{DATE} \\
\hline
\mathit{name?} \notin \mathit{known} \\
\mathit{birthday}' = \mathit{birthday} \\
\qquad\qquad \cup \{\mathit{name?} \mapsto \mathit{date?}\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{AddBirthday1}} \\
\hline
\Delta \mathit{BirthdayBook1} \\
\mathit{name?} : \mathit{NAME} \\
\mathit{date?} : \mathit{DATE} \\
\hline
\forall\, i : 1 \mathinner{..} \mathit{hwm} \bullet \mathit{name?} \neq \mathit{names}(i) \\
\mathit{hwm}' = \mathit{hwm} + 1 \\
\mathit{names}' = \mathit{names} \oplus \{\mathit{hwm}' \mapsto \mathit{name?}\} \\
\mathit{dates}' = \mathit{dates} \oplus \{\mathit{hwm}' \mapsto \mathit{date?}\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Abs}} \\
\hline
\mathit{BirthdayBook} \\
\mathit{BirthdayBook1} \\
\hline
\mathit{known} = \{\, i : 1 \mathinner{..} \mathit{hwm} \bullet \mathit{names}(i) \,\} \\
\forall\, i : 1 \mathinner{..} \mathit{hwm} \bullet \\
\qquad \mathit{birthday}(\mathit{names}(i)) = \mathit{dates}(i) \\
\hline
\end{array}
$$

(a) Explain, with reference to the above schemas, what is meant by the terms *direct refinement* and *deferred refinement*, as used by Spivey in discussing data refinement of the birthday book specification. [4 marks]

(b) Write the specification for *AddBirthday*1, the data refinement of *AddBirthday*, you would obtain using *deferred refinement.* [5 marks]

(c) Show that Spivey's version of *AddBirthday*1 is a refinement of the version you wrote in part (b). [5 marks]

Question 3. **[14 marks]**

(a) State Dijkstra's *healthiness conditions* for programs defined by weakest precondition predicate transformers. [4 marks]

(b) All but one of Dijkstra's healthiness conditions are broken by some of the constructs used in the refinement calculus.

Identify the healthiness conditions which are broken, and give an example of a construct in the refinement calculus which breaks each such condition. [2 marks]

(c) Define the weakest precondition semantics for **if** $[\!]_i\, G_i \rightarrow S_i$ **fi**, as in Dijkstra's guarded command language, and prove that the resulting predicate transformer is *monotonic* with respect to implication, assuming each S_i is monotonic. [4 marks]

(d) Give sufficient conditions (not involving the weakest precondition of the **do** statement) to ensure that $P \Rightarrow wp(\mathbf{do}\ [\!]_i\, G_i \rightarrow S_i\ \ \mathbf{od}, R)$.

Explain the practical significance of this result. [4 marks]

Question 4. **[12 marks]**

Suppose we define a parallel composition operator ($||$), defined in terms of specification statements as follows:

$$w_1\!: \left[P_1\ ,\ R_1 \right]\ \ ||\ \ w_2\!: \left[P_2\ ,\ R_2 \right]\ \equiv\ w_1 \cup w_2\!: \left[P_1\ \wedge\ P_2\ ,\ R_1\ \wedge\ R_2 \right]$$

Prove the following:

(a) $S\ ||$ **abort** $\equiv$ **abort**, for any specification statement S. [2 marks]

(b) $S\ ||$ **magic** $\equiv$ **magic**, for any specification statement S, provided that S always terminates. [2 marks]

(c) $x := e_1\ ||\ x := e_2$ is equivalent to $x := e_1$ if $e_1 = e_2$, and **magic** otherwise. [2 marks]

(d) $(S\ [\!]\ T)\ ||\ U\ \equiv\ (S\ ||\ U)\ [\!]\ (T\ ||\ U)$, for any specification statements S, T and U, where S and T have the same frame. [6 marks]

Question 5. **[22 marks]**

The following specification statement specifies a program to determine whether a sequence, A, is a palindrome:

$$z\mathbf{:}\left[\, true \;,\; z \equiv pal(A) \,\right] \tag{1}$$

where: $pal(A) \;\widehat{=}\; (\forall\, i : 1..\#A \text{ div } 2 \bullet A[i] = A[\#A{+}1{-}i])$, and

$\#A$ is the length of A

(a) Suppose you decide to implement this specification by counting the number of pairs (i, j), with $j = \#A{+}1{-}i$, for which $A[i] = A[j]$, and then comparing this count with $\#A$. So you refine (1) to:

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

Show that this program is a valid refinement of specification statement (1). 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. [10 marks]

(d) The above implementation of (1) is inefficient, because it compares every element $A[i]$ with $A[\#A{+}1{-}i]$, even if an unequal pair has already been found. It also compares every pair of elements twice. A better implementation would stop as soon as it is known whether A is a palindrome or not. Give a loop invariant which would lead to this implementation, and explain briefly how using this invariant would affect the resulting program. [4 marks]

Question 6. **[20 marks]**

Consider the problem of finding the minimum element in a non-empty array segment $A[1..n]$, specified as follows:

$$r\colon \left[\, 1 \le n \le \#A \;,\; r = min(A[1..n]) \,\right]$$

The following is a program which implements this specification:

$$
\begin{array}{l}
|[\;\; \textbf{var}\ k, m \bullet \\
\quad k, m \;:=\; 1, A[1]; \\
\quad \textbf{do}\ k \ne n \rightarrow \\
\qquad \textbf{if}\;\; A[k+1] \le m \rightarrow m \;:=\; A[k+1] \\
\qquad [\!]\;\; A[k+1] \ge m \rightarrow \textbf{skip} \\
\qquad \textbf{fi}; \\
\qquad k \;:=\; k+1 \\
\quad \textbf{od}; \\
\quad r \;:=\; m \\
]|
\end{array}
$$

Suppose you wish to transform this program into one which records the position, p, of a minimal element, rather than recording the minimum value.

(a) Explain how this data refinement can be performed using the *auxiliary variables* technique, using the coupling invariant:

$$I \;\widehat{=}\; m = A[p] \;\wedge\; 1 \le p \le k$$

Show the resulting program at each step. [10 marks]

(b) Show that the result of applying the auxiliary variables technique is a data refinement of the original program, by showing that if:

(i) $Sa \preceq_{I,\epsilon,c} Sac$,

(ii) $Sac \;\sqsubseteq\; Sac'$, and

(iii) $Sac' \preceq_{I,a,\epsilon} Sc$

then $Sa \preceq_{I,a,c} Sc$

where Sa involves a but not c, Sc involves c but not a, and Sac and Sac' involve both a and c, and $Sa \preceq_{I,a,c} Sc$ iff, for all postconditions R,

$$(\exists\, a \bullet I \;\wedge\; wp(Sa, R)) \Rightarrow wp(Sc, (\exists\, a \bullet I \;\wedge\; R))$$ [10 marks]

****