TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2006

MID-YEAR

COMP 426

Formal Software Development

Time Allowed: 3 Hours

Instructions: Candidates should attempt **all THREE** questions.
This exam will be marked out of 100.
Foreign language translation dictionaries are allowed.

Question 1. Program Verification [38 marks]

Consider the following program, which sets r to the number of locations in the array segment $a[1..n]$ whose value occurs again later in $a[1..n]$:

$\{$**pre:** $n \geq 0\}$
$r := 0;$
$i := 1;$
while $i \leq n$ **do** $\{$**inv:** $1 \leq i \leq n+1 \wedge r = dups(a[1\,..\,n], i-1)\}$
  $j := i + 1;$
  $s := r;$
  while $j \leq n \wedge s = r$ **do**
    if $a[i] = a[j]$ **then**
      $r := r + 1$
    fi;
    $j := j + 1$
  od;
  $i := i + 1$
od
$\{$**post:** $r = dups(a[1\,..\,n], n)\}$

The loop invariant and postcondition are stated in terms of the function *dups* defined as:

$$dups(a[1\,..\,n], k) = \#\{\ p : 1\,..\,k \mid \exists\, q : p{+}1\,..\,n \bullet a[p] = a[q]\ \}$$

where $\#s$ denotes the size of the set s.

Note that if a value occurs more than twice, then all but one occurrence is counted; for example, if a is a four element array with $a[1] = a[2] = a[3] = a[4]$, then $dups(a[1\,..\,4], 4) = 3$.

(a) [6 marks] Show that the given loop invariant (**inv**) holds on entry to the outer loop, and guarantees the given postcondition (**post**) on exit from the outer loop.

(b) [12 marks] Give a loop invariant that could be used to verify the inner loop. Explain what your invariant says about the program variables, and give a brief justification for the correctness of the loop in terms of this invariant.

This justification should give an informal argument to show that your invariant holds on entry to the inner loop, that it is preserved by the body of the inner loop, and that it guarantees the appropriate postcondition on exit from the inner loop.

(c) [10 marks] Explain briefly how you would construct a formal verification of this program using **(i)** Floyd's inductive assertions method, and **(ii)** Hoare logic.

In answering this question, you should emphasise the main differences between the two approaches and how this affects the *structure* of the proof. You are not required to construct the proofs, except as needed to illustrate your answer, and you are not required to address termination.

(d) [10 marks] The variable s was introduced solely to allow the program to be expressed using the simple nested control structures available in the language of while programs. If we added an **exit** statement to the language, which caused the program to exit from the closest enclosing loop (like `break` in Java), we could express the program more simply as follows:

$\{$**pre:** $n \geq 0\}$
$r := 0;$
$i := 1;$
while $i \leq n$ **do** $\{$**inv:** $1 \leq i \leq n+1 \wedge r = dups(a[1..i-1])\}$
  $j := i + 1;$
  while $j \leq n$ **do**
    if $a[i] = a[j]$ **then**
      $r := r + 1;$
      exit
    fi;
    $j := j + 1$
  od;
  $i := i + 1$
od
$\{$**post:** $r = dups(a[1..n])\}$

Discuss the effect that this modification would have on the verification of the program.

In answering this question, you should:

- Show the effect that this modification has on the invariant for the inner loop and your informal justification of the correctness of the inner loop (as given in part **(b)**);
- Explain the effect that the modification would have on a verification constructed using Floyd's method (as described in part **(c)**); and
- Explain the effect that the modification would have on a verification constructed using Hoare logic (as described in part **(c)**).

Question 2. Semantics [34 marks]

In this course, we have seen a number of mathematical structures that can be used to represent the semantics of programs and specifications, including functions, relations and predicate transformers, where functions and relations may be partial or total and may or may not include special *error* values, and predicate transformers may be subject to various *healthiness* conditions.

(a) [10 marks] Explain briefly the kinds of behaviour that can be represented (and distinguished) **(i)** using functions and **(ii)** using relations, and how this is affected by assumptions about totality and the existence of special *error* values.

(b) [8 marks] Explain briefly the healthiness conditions that Dijkstra imposed on the weakest precondition predicate transformers used to represent the semantics of programs in his guarded commands language, and why he claimed that each was necessary.

(c) [8 marks] Most of Dijkstra's healthiness conditions are dropped in the refinement calculus. For each of Dijkstra's healthiness conditions, **either** explain why it is retained in the refinement calculus, **or** explain what behaviours are admitted by dropping the condition and why it is desirable to allow such behaviours in a wide-spectrum language.

(d) [8 marks] Dijkstra defines wp for a do statement $DO = \mathbf{do}\ G \rightarrow S\ \mathbf{od}$ as:

$$wp(DO)(R) = \exists k \in \mathbb{N} \bullet H_k(R)$$

where H_k is defined inductively as:

$$H_0(R) = \neg G \wedge R$$
$$H_{k+1}(R) = G \wedge wp(S)(H_k(R)) \vee H_0(R)$$

Show that do statements preserve monotonicity (i.e. show that if S is monotonic, then so is DO).

Question 3. Program Refinement and Equivalence [28 marks]

The refinement relation used in the refinement calculus is defined, for programs S and T in the wide spectrum language, as follows:

$$S \sqsubseteq T \mathrel{\widehat{=}} wp(S)(R) \Rightarrow wp(T)(R), \text{ for all postconditions } R$$

(a) [5 marks] Show that refinement is *transitive*, and explain why this property is important.

(b) [5 marks] Show that **if** statements are *monotonic* with respect to refinement (i.e. show that if $S \sqsubseteq S'$, then $\textbf{if}\ G \to S \,[\!]\, H \to T\ \textbf{fi} \sqsubseteq \textbf{if}\ G \to S' \,[\!]\, H \to T\ \textbf{fi}$.

(c) [6 marks] Prove that the following law holds for all specification statements $w : \left[\, P \;/\; Q \,\right]$ and guards G and H such that $P \Rightarrow G \vee H$:

$$w : \left[\, P \;/\; Q \,\right] \sqsubseteq \textbf{if}\ G \to w : \left[\, G \wedge P \;/\; Q \,\right] \,[\!]\, H \to w : \left[\, H \wedge P \;/\; Q \,\right]\ \textbf{fi}$$

(d) [4 marks] Prove that the following law holds for all S, T and U:

$$\textbf{if}\ G \to S\,;U \,[\!]\, H \to T\,;U\ \textbf{fi} \;\equiv\; \textbf{if}\ G \to S \,[\!]\, H \to T\ \textbf{fi}\,;U$$

You may answer this question **either** using relational semantics, **or** using weakest precondition semantics and assuming that $X \equiv Y$ means $X \sqsubseteq Y$ and $Y \sqsubseteq X$. You do not need to state the semantic definitions used, as long as the definitions you are using are clear from your answer.

(e) [8 marks] Consider the following proposed law, where $[\!]$ is interpreted as demonic choice:

$$S\,;(T \,[\!]\, U) \;\equiv\; (S\,;T) \,[\!]\, (S\,;U)$$

(i) Prove that this law holds in a relational semantic model.

(ii) Explain why the law does not hold in the weakest precondition semantic model used in the refinement calculus, and provide a counter-example.

****