TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA

UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2011

END-OF-YEAR

SWEN 224
Formal Foundations of Programming

Time Allowed: 3 Hours

Instructions:

- Answer all **four** questions.
- The exam will be marked out of one hundred and eighty (180).
- Calculators ARE NOT ALLOWED.
- Non-electronic foreign language dictionaries are allowed.
- No other reference material is allowed.

Question 1. Assertions and Verification [60 marks]

(a) [15 marks : 5 for each part] Write an *assertion* formalising each of the following statements, where A is an array of size N. You may use either JML or ordinary mathematical notation.

(i) All elements of A are different.

(ii) A contains exactly one occurrence of z.

(iii) Between any two occurrences of z in A, there is at least one occurrence of y.

(b) [15 marks : 5 for each part] For each of the following correctness assertions, write down the *verification condition(s)* that must hold in order for the correctness assertion to be valid, and give a brief explanation of why these verification conditions hold.

(i) $\{ k = 0 \}\, s := 0\, \{ s = \sum_{i=0}^{k-1} A[i] \}$

(ii) $\{ x \leq y \}$ **if** $x > z$ **then** $x := z$ **else skip fi** $\{ x \leq y \wedge x \leq z \}$

(iii) $\{ 0 \leq k < n-1 \wedge s = \sum_{i=0}^{k-1} A[i] \}\, k := k+1;\, s := s+A[k]\, \{ 0 \leq k < n \wedge s = \sum_{i=0}^{k-1} A[i] \}$

(c) Consider the following Java method, which counts the number of times 0 occurs in an integer array A.

```
//@ requires A != null;
//@ ensures \result == (\num_of int k; 0 <= k && k < A.length; A[k] == 0);
int countZeroes(int[] A) {
  int i = 0;
  int c = 0;
  while (i < A.length) {
    if ( A[i] == 0 ) c = c + 1;
    i = i + 1;
  }
  return c;
}
```

(i) [4 marks] The `requires` and `ensures` annotations give the pre and postconditions for the method. What are the pre and postconditions for the loop?

(ii) [5 marks] Give a loop invariant that can be used to verify the loop in this method.

(iii) [15 marks] State the three verification conditions that must be proved in order to show that the loop is partially correct, and give a brief argument to show that each of them holds. (You may use ordinary mathematical notation instead of JML if you prefer.)

(iv) [6 marks] Give a brief argument to show that the method terminates properly, i.e. that it does not give a run-time error or exception and does not loop forever. (You may ignore the possibility of arithmetic overflow.)

Question 2. Alloy

[20 marks]

Consider the following Alloy model for Nondeterministic Finite Acceptors:

```
sig Symbol {}

sig State {}

sig NFA {
  initial: State,
  next: State -> Symbol -> State,
  final: set State
}

sig Config {
  state: State,
  input: seq Symbol
}

pred move[m: NFA, c1, c2: Config] {
  c2.state = m.next[c1.state][c1.input.first]  &&
  c2.input = c1.input.rest
}

pred accepts[m: NFA, s: seq Symbol] {
  some ss: seq Config |
    ss.first.state = m.initial && ss.last.state in m.final &&
    (all i: ss.inds-#ss | move[m, ss[i], ss[i+1]])
}
```

(a) [3 marks] Write a predicate to determine whether an NFA has no final state.

(b) [3 marks] Write a predicate to determine whether an NFA has at least two final states.

(c) [3 marks] Write a predicate to determine whether an NFA is deterministic.

(d) [3 marks] Write a predicate to determine whether an NFA accepts the empty string.

(e) [3 marks] Write a predicate to determine whether the languages accepted by two NFAs have any strings in common.

(f) [5 marks] Write a predicate to determine whether two NFAs has any common states; i.e. states that are reachable from the initial states of both machines.

Question 3. Regular Languages [55 marks]

(a) [8 marks : 4 for each part] Write a *Regular Expression* or *Regular Grammar* to describe each of the following languages, over the alphabet $\{a, b, c\}$:

(i) All strings in which all a's come before all b's and all c's.

(ii) All strings of even length in which all a's come before all b's.

(b) Consider the NFA $M = (Q, q_I, A, N, F)$, where:

$$
\begin{aligned}
Q &= \{1, 2, 3, 4, 5\} \\
q_I &= 1 \\
A &= \{\text{a}, \text{b}\} \\
N &= \{(1, a, 1), (1, a, 2), (1, b, 1), (1, b, 3), (2, a, 4), (2, b, 2), \\
&\quad (3, a, 3), (3, b, 5), (4, a, 4), (4, b, 4), (5, a, 5), (5, b, 5)\} \\
F &= \{4, 5\}
\end{aligned}
$$

(i) [4 marks] Draw a transition diagram for M.

(ii) [4 marks] Show the sequence of configurations that M passes through in accepting the input *ababa*.

Note that you should show *all* states that M may be in after accepting part of the input.

(iii) [4 marks] Write a regular expression describing the language accepted by M.

(iv) [8 marks] Draw a transition diagram for the DFA obtained by applying the *subset construction* to M. Show the correspondence between states of the DFA and those of the NFA. You only need to show reachable states.

(c) [10 marks] Given an NFA $M = (Q, q_I, A, N, F)$, show how to construct an NFA, $M' = (Q', q_I', A', N', F')$, which accepts the language consisting of all strings in $\mathcal{L}(M)$ enclosed in a pair of a's. For example, if $\mathcal{L}(M) = \{a, aa, aaa\}$, then $\mathcal{L}(M') = \{aaa, aaaa, aaaaa\}$, and if $\mathcal{L}(M) = \{\lambda, b, c, bcb\}$, then $\mathcal{L}(M') = \{aa, aba, aca, abcba\}$.

Give a brief argument to show that the resulting NFA does in fact accept the required language.

(d) [6 marks] Draw a transition diagram for an NFA$_\epsilon$ that accepts the language defined by the regular expression $a^*(b \mid c)^* \mid (a \mid b)^*c^*$.

(e) [3 marks] Explain briefly why allowing null transition makes it easier to construct an NFA from a regular expression.

(f) [8 marks] Prove that, for any regular expressions x, y and z, $x(y \mid x) = xy \mid xz$.

Question 4. Context-Free Languages [45 marks]

(a) [10 marks : 5 for each part] Write a *Context Free Grammar* to describe each of the following languages. You are not required to give a full formal definition of these grammars — just write the list of rules.

(i) All strings consisting of one or more d's, optionally followed by an a and one or more further d's. For example, *d*, *ddd*, *dad* and *ddadd* are in this language, but *add*, *dda* and *dada* are not.

(ii) All strings formed by concatenating two non-empty palindromes over $\{a, b, c\}$, where a palindrome is a strings that reads the same forwards and backwards (e.g. *a*, *aa*, *aba*, *abcabccbacba*). Thus, *aa*, *abaaba*, *abab*, *bbaaa* and *abcabccbacbaaaa* are in this language, but *a*, *aba*, *abbc* and *abcabc* are not.

(b) Consider the following grammar, where "!", "\$", "*a*", *b*", "(" and ")" are terminal symbols:

$$
\begin{array}{ll}
S \to T\,! \mid T\,T\,!\,! & (1,2) \\
T \to T\,\$\,T \mid U & (3,4) \\
U \to a \mid b \mid (\,T\,) & (5,6,7)
\end{array}
$$

(i) [3 marks] Construct a parse tree for "$a\,\$\,b\,!$".

Write the number of the rule applied beside each nonterminal in the parse tree; likewise for parts **(ii)** and **(iii)**.

(ii) [4 marks] Construct a parse tree for "$(\,a\,\$\,b\,)\,a\,!\,!$".

(iii) [6 marks] Demonstrate that the grammar is ambiguous by drawing two differrent parse trees for "$a\,\$\,b\,\$\,a\,!$".

(iv) [6 marks] Write an equivalent, non-ambiguous grammar, treating "\$" as *left-associative*.

(v) [8 marks] Write an equivalent LL(1) grammar, and show that it is LL(1).

(c) [8 marks] Prove that the union of two context-free languages is context-free.

Hint: Recall that a language is context-free if and only if it can be defined using a context-free grammar.

****