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

EXAMINATIONS — 2009

MID-TERM TEST

COMP 202 / SWEN 202
Formal Methods of Computer Science / Formal Foundations of Software Engineering

Time Allowed: 90 minutes

Instructions: There are **five** (5) questions.

Answer **all** the questions.

The exam will be marked out of **eighty** (80).

Calculators ARE NOT ALLOWED.

Non-electronic foreign language dictionaries are allowed.

No other reference material is allowed.

Question 1. [20 marks]

The following Alloy provides a model of family relations.

```
abstract sig Person {
  mother: lone Woman,
  father: lone Man,
}
sig Man, Woman extends Person {}
```

(a) [8 marks]

Consider the following instance:

$$\texttt{Man} = \{M0, M1\} \qquad \texttt{Woman} = \{W0, W1, W2\}$$

$$\texttt{father} = \begin{array}{|c|c|}\hline M1 & M0 \\ \hline W2 & M1 \\ \hline \end{array} \qquad \texttt{mother} = \begin{array}{|c|c|}\hline M1 & W0 \\ \hline W1 & W0 \\ \hline \end{array}$$

(i) [2 marks] Draw a visualisation (i.e. a graph representation as Alloy would give) of this instance.

(ii) [2 marks] Compute `mother.~mother`

(iii) [2 marks] Compute `Man <: (father + mother).`

(iv) [2 marks] Compute `^father`

(b) [12 marks]

(i) [2 marks] Provide a run command that allows you to show instances with at least one person (that is, in which set `Person` is not empty).

(ii) [3 marks] Write an Alloy function that takes a person as argument and returns all his or her descendants; i.e. children, grandchildren, great-grandchildren, etc.

(iii) [2 marks] Write a fact that ensures that no person is a descendant of itself.

(iv) [2 marks] Write an Alloy predicate that takes two persons as arguments and is true if the two have at least one child in common.

(v) [3 marks] Provide an Alloy command to check that, forall persons that have a child in common, one of them must be a man and one of them a woman. Is this assertion true for the model given above?

Question 2. [20 marks]

Consider the following Alloy model of a file system:

```
sig Directory {}

sig FileSystem {
  root: Directory,
  dirs: set (Directory-root),
  parent: dirs -> one (dirs + root)
}

pred init[fs: FileSystem] {
  no fs.dirs
}
```

(a) [2 marks]

Adding a fact to the above specification requiring the root directory to be an ancestor (i.e. a parent or a parent of a parent etc.) of itself would make the model inconsistent. Explain, with reference to this example, what it means for a model to be inconsistent.

(b) [4 marks]

Provide an invariant (a predicate called `inv`) that is true for file systems in which the root directory is an ancestor (i.e. a parent or a parent of a parent etc.) of all non-root directories in the file system.

(c) [2 marks]

The given predicate `init` describes the initial states of the file system. What can you say about the `parent` relation of initial file systems?

(d) [3 marks]

Write an Alloy command to check that initial file systems satisfy the invariant provided in **(b)**. Is this assertion true for the model given above?

(e) [3 marks]

What does it mean for an operation to preserve an invariant?

(f) [6 marks]

Provide an operation that models adding a new directory to a given directory of a file system. Make sure your operation preserves the invariant provided in **(b)**.

Question 3. [14 marks]

Consider the following JML annotated Java program:

```
public class Person
{
  //@ invariant height >= 0;
  private /*@ spec_public @*/ int height;

  //@ ensures height == \old(height) + distance;
  public void grow(int distance) {
    height += distance;
  }

  public /*@ pure @*/ int getHeight() {
    return height;
  }

  public static void main(String[] args) {
    Person p = new Person();
    p.grow(30);
    System.out.println(p.getHeight());
  }
}
```

(a) [1 mark]

The program compiles using `jmlc` without problems. What will happen if you execute it using `jmlrac`?

(b) [3 marks]

Provide a new `main` method that shows that the `grow` method does not preserve class `Person`'s invariant. What will happen if you execute your new `main` method using `jmlrac`?

(c) [4 marks]

Provide a JML annotation that ensures that the `grow` method preserves the invariant. What will happen if you execute your main method from part **(b)** with this modified program using `jmlrac`?

(d) [3 marks]

When does a *postcondition* for a method need to hold? Under which circumstances is the postcondition not required to hold?

(e) [3 marks]

What is a *pure* method? Why can only pure methods be used in JML specifications?

Question 4. [6 marks]

Do the following methods correctly implement their specification? Give a brief explanation why you think they do or do not.

(a) [2 marks]

```
//@ requires true;
//@ ensures \result == 0 || \result == 1;
int foo() {
  return 0;
}
```

(b) [2 marks]

```
//@ requires x != y;
//@ ensures \result == x || \result == y;
//@ ensures \result > x || \result > y;
int bar(int x, int y) {
  if (x >= y) return x;
  return y;
}
```

(c) [2 marks]

```
//@ requires false;
//@ ensures true;
int baz() {
  throw new IllegalArgumentException();
}
```

Question 5. [20 marks]

Consider a Java method:

```
boolean equivalent(int[] a, int[] b) {
  int i = 0;
  while (i < a.length && a[i] == b[i]) {
    i++;
  }
  return i == a.length;
}
```

This method takes as input two integer arrays of the same length and determines whether the given arrays are equivalent, that is, contain the same integers in exactly the same order.

(a) [4 marks]

Give a JML specification (precondition and postcondition) for this method.

(b) [6 marks]

Provide a loop invariant that may be used in the verification of the above implementation of the method.

(c) [10 marks]

Give an argument (informal proof), using your loop invariant from part **(b)**, to show that the method correctly implements its specification; i.e. show that:

(i) [2 marks] The loop invariant holds on entry to the loop.

(ii) [3 marks] The loop invariant is preserved by the loop body.

(iii) [5 marks] The postcondition of the method holds when the loop exits with the loop invariant true.

****