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

Name: ..... ID: .....

EXAMINATIONS — 2016

TRIMESTER 2

SWEN224

Software Correctness

Time Allowed: ONE HOUR

Instructions: Answer all questions
All questions are of equal value

Answer all questions in the boxes provided.
Every box requires an answer.
If additional space is required you may use a separate answer booklet.

No calculators permitted.
Non-electronic Foreign language to English dictionaries are allowed.

Question	Topic	Marks
1.	Static Analysis	20
2.	Specification	20
3.	Loop Invariants	20
	Total	60

Question 1. Static Analysis [20 marks]

(a) [3 marks] Static analyses operate at *compile time*. Briefly, discuss the disadvantages of this, compared with techniques which operate at *run time*.

(b) This question is concerned with *definite assignment* in Java.

(i) [4 marks] Briefly, explain what definite assignment means in Java. You may illustrate your answer with examples as necessary.

Consider the following method in Java:

```
1      int max(int[] xs) {
2          if (xs.length == 0) {
3              throw new IllegalArgumentException("invalid array");
4          } else {
5              int r;
6              for (int i = 0; i != xs.length; ++i) {
7                  if (i == 0 || xs[i] > r) {
8                      r = xs[i];
9                  }
10             }
11             return r;
12         }
13     }
```

(ii) [2 marks] Briefly, explain why the above method fails definite assignment.

(iii) [5 marks] Like most static analyses, definite assignment is said to be *conservative*. Explain what this means using the above `max(int[])` method to illustrate.

(c) [6 marks] For each *parameter*, *return* and *field* in the following program, insert `@NonNull` or `@Nullable` annotations (where appropriate) by writing in the box.

```
1  public class Tree {
2      private Object data; // Every tree node has a data item
3
4      private Tree left;    // Some trees have a left child
5
6      private Tree right;   // Some trees have a right child
7
8      public Tree(Object data) {
9          if (data == null) {
10             throw new IllegalArgumentException("Data cannot be null");
11         }
12         this.data = data;
13
14         this.left = null;
15
16         this.right = null;
17     }
18
19     public Object getData() {
20         return data;
21     }
22
23     public Tree getLeftChild() {
24         return left;
25     }
26
27     public Tree getRightChild() {
28         return right;
29     }
30
31     public void appendLeft(Object item) {
32         if (left == null) {
33             left = new Tree(item);
34         } else {
35             left.appendLeft(item);
36         }
37     }
38
39     public void appendRight(Object item) {
40         if (right == null) {
41             right = new Tree(item);
42         } else {
43             left.appendRight(item);
44         }
45     }
46 }
```

Name: . . . . . ID: . . . . . .

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 2. Specification [20 marks]

(a) For each of the following, provide one set of parameter values which meet the precondition and one set which does not.

(i) [2 marks]

```
function decrement(int value) -> (int r)
requires value > 0:
    //...
```

Meets precondition:

Does not meet precondition:

(ii) [2 marks]

```
function copy(int[] xs, int start, int[] ys) -> (int[] r)
requires (start+|xs|) < |ys| && start >= 0:
    //...
```

Meets precondition:

Does not meet precondition:

(iii) [2 marks]

```
function findNegative(int[] xs) -> (int r)
requires some { i in 0..|xs| | xs[i] < 0 }:
    //...
```

Meets precondition:

Does not meet precondition:

(iv) [2 marks]

```
function bitwiseAnd(int[] lhs, int[] rhs) -> (int[] r)
requires all { i in 0..|lhs| | 0 <= lhs[i] && lhs[i] <= 255 }
requires all { i in 0..|rhs| | 0 <= rhs[i] && rhs[i] <= 255 }:
    //...
```

Meets precondition:

Does not meet precondition:

(b) Consider the following implementation for the function `find()`:

```
1 function find(int n, int[] items, int i) -> (int r):
2     if i == |items|:
3         return i
4     else if items[i] == n:
5         return i
6     else:
7         return find(n,items,i+1)
```

(i) [4 marks] Briefly, describe in your own words what function `find()` does.

(ii) [8 marks] Provide an appropriate specification for function `find()`.

Question 3. Loop Invariants [20 marks]

(a) [6 marks] Briefly, discuss what a *loop invariant* is and explain the *three rules* of loop invariants.

Consider the following implementation for the function `count()`:

```
1 function count(int from, int to) -> (int r)
2 requires from < to:
3     //
4     int i = from
5     //
6     while i < to where i >= 0:
7         i = i + 1
8     //
9     return i
```

(b) [3 marks] Briefly, discuss whether or not the given loop invariant is correct.

Consider the following implementation for the function `rotate()`:

```
1  function rotate(int[] xs) -> (int[] ys)
2  requires |xs| > 0:
3  ensures |xs| == |ys|
4  ensures all { k in 1 .. |xs| | xs[k] == ys[k-1] }
5  ensures ys[|xs|-1] == xs[0]:
6      //
7      int first = xs[0]
8      int[] ys = xs
9      int i = 1
10     //
11     while i < |xs|:
12         ys[i-1] = xs[i]
13         i = i + 1
14     //
15     ys[|xs|-1] = first
16     //
17     return ys
```

(i) [5 marks] Briefly, describe in your own words what function `rotate()` does.

(ii) [6 marks] Provide an appropriate *loop invariant* for function `rotate()`.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Name: ..... ID: .....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

****