

Te Herenga Waka—Victoria University of Wellington

EXAMINATIONS – 2022

TRIMESTER 1

<p>SWEN 326</p> <p>SAFETY-CRITICAL SYSTEMS</p>
--

Time Allowed: TWO HOURS

CLOSED BOOK

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

Instructions: Answer all questions

Question	Topic	Marks
1.	Risk, Hazards and Failure	30
2.	Testing	30
3.	Static Analysis	30
4.	Design Validation	30
Total		120

1. Risk, Hazards and Failure

(30 marks)

(a) Consider the following description of a system for controlling a *model rocket*:

“The rocket is operated by a *software controller*. If the *navigation sensor* detects that the trajectory is too low, the engine is *turned off* and the parachute is *deployed*. Under no circumstance should the rocket be allowed to travel at speed towards the ground.”

i. (2 marks) Following the terminology of IEC61508, identify the *Equipment Under Control* for the rocket system.

ii. (2 marks) Identify an important *hazard* for the rocket system.

iii. (2 marks) Briefly, discuss what is required to estimate the *risk* posed by the above hazard.

iv. (2 marks) Following the terminology of IEC61508, identify which part of the system is relied upon to ensure *functional safety*.

v. (2 marks) Briefly, discuss how the above hazard is *mitigated* in the system.

(Question 1 continued on next page)

(Question 1 continued)

- vi. **(4 marks)** Under IEC61508 there is always *residual risk*. Briefly, discuss what this means with respect to the rocket system.

- (b) **(4 marks)** Briefly, discuss why the possibility of a *multiple component failure* can be particularly problematic.

- (c) **(4 marks)** Rule 3 from the “*Power-of-Ten*” coding guidelines prohibits the use of “*dynamic memory allocation after initialization*”. Briefly, discuss the motivation behind this rule.

(Question 1 continued on next page)

(Question 1 continued)

- (d) **(4 marks)** The SWEN326 Java Coding Standard requires that “*JavaDoc errors must be enabled*”. Briefly, discuss the pros/cons of this requirement.

- (e) **(4 marks)** The SWEN326 Java Coding Standard used for the steam boiler assignment requires that “Eclipse’s non-null analysis is enabled”. Briefly, discuss the pros/cons of this requirement.

SPARE PAGE FOR EXTRA ANSWERS

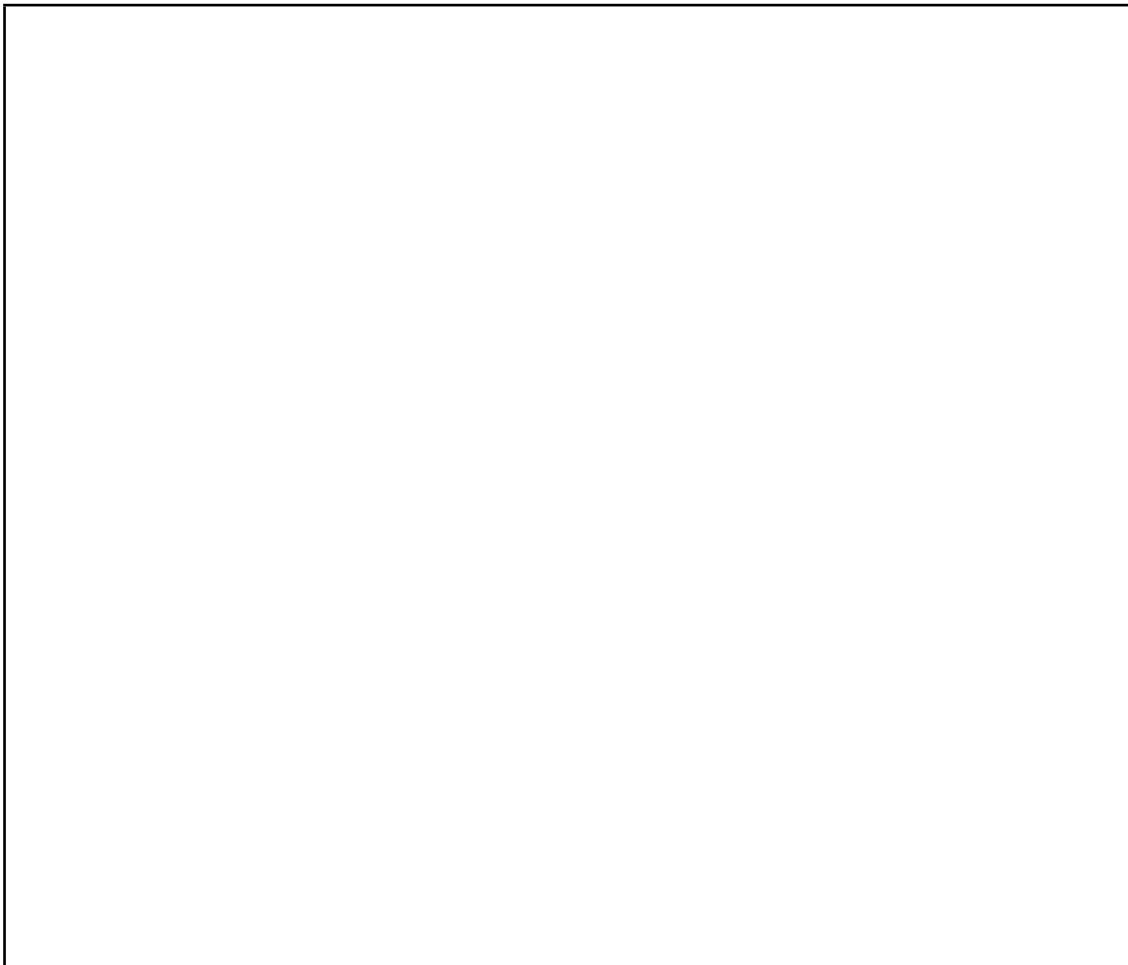
Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

2. Testing

(30 marks)

Consider the following Java method which compiles without error:

```
1   public int[] zip(int[] lhs, int[] rhs) {
2       if(lhs == null || rhs == null) {
3           return null;
4       } else if(lhs.length != rhs.length) {
5           return null;
6       } else {
7           int[] rs = new int[lhs.length + rhs.length];
8           // Merge like a zip!
9           for(int i=0; i!=rs.length; ++i) {
10              if(i%2 == 0) { rs[i] = lhs[i / 2]; }
11              else { rs[i] = rhs[i / 2]; }
12          }
13          // Done
14          return rs;
15      } }
```

(a) (10 marks) Draw the *control-flow graph* for the `zip()` method.

(Question 2 continued on next page)

(Question 2 continued)

(b) **(6 marks)** Write three JUnit tests which achieve 100% *branch coverage* of `zip()`.

```
1     @Test public void test_1() {  
2  
3  
4  
5  
6     }
```

```
1     @Test public void test_2() {  
2  
3  
4  
5  
6     }
```

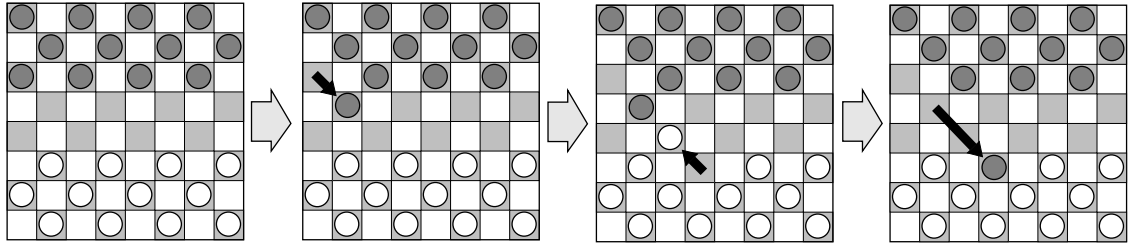
```
1     @Test public void test_3() {  
2  
3  
4  
5  
6     }
```

(c) **(3 marks)** Briefly, discuss whether additional tests are required to achieve 100% *Modified Condition/Decision Coverage (MC/DC)* for the `zip()` method.

(Question 2 continued on next page)

(Question 2 continued)

(d) In the game of *drafts* there are two players. The first player uses *black pieces* and the second player uses *white pieces*. The *board* consists of 64 squares arranged in an 8x8 grid. Each square holds at most one piece. Players take turns *moving* their pieces one square at a time along the diagonals of the board. Players can *capture* their opponent's pieces by jumping over them into an empty square. The following illustrates the first three moves of a game:



Suppose we have a program which implements the rules of drafts.

i. **(5 marks)** Briefly, discuss what a *test oracle* for this program might look like.

ii. **(6 marks)** Suppose we test our program by generating random games consisting of *at most two moves* by each player. Briefly, discuss the pros / cons of this approach:

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.

3. Static Analysis

(30 marks)

(a) This question is concerned with *static analysis*.

i. (3 marks) Briefly, discuss what is meant by the term static analysis.

ii. (3 marks) Briefly, discuss the difference between *compile-time* and *run-time*.

(b) This question is concerned with *non-null* analysis.

i. (2 marks) Briefly, discuss what the `@Nullable` annotation means.

ii. (2 marks) Briefly, discuss what “*non-null by default*” means for a non-null analysis.

iii. (2 marks) Briefly, discuss what a *false negative* means with respect to a non-null analysis.

(Question 3 continued on next page)

(Question 3 continued)

- iv. **(9 marks)** For each *parameter*, *return* and *field* in the following program, insert @NonNull or @Nullable annotations (as appropriate) by writing in the box. You should not assume non-null by default.

```

1  class Point {
2
3      public final int x, y;
4
5      public Point(int x, int y) { this.x = x; this.y = y; }
6  }
7
8  interface Shape { boolean contains(Point p); }
9
10 class Rectangle implements Shape {
11
12     private final Point tl;
13
14     private final Point br;
15
16     public Rectangle(int x, int y, int width, int height) {
17         tl = new Point(x,y);
18         br = new Point(x+width-1,y+height-1);
19     }
20
21     public boolean contains(Point p) {
22         return (tl.x <= p.x) && (p.x <= br.x) &&
23             (tl.y <= p.y) && (p.y <= br.y);
24     }
25 }
26
27 class Union implements Shape {
28
29     private final Shape l;
30
31     private final Shape r;
32
33     public Union(Shape lhs, Shape rhs) { l = lhs; r = rhs; }
34
35     public boolean contains(Point p) {
36         if(p == null) { return false; }
37         else if(l==null && r==null) { return false; }
38         else if(l==null && r!=null) { return r.contains(p); }
39         else if(l!=null && r==null) { return l.contains(p); }
40         return l.contains(p) || r.contains(p);
41     }
42 }

```

(Question 3 continued on next page)

(Question 3 continued)

- v. **(3 marks)** The `assert` statement can help with foreign code that has not been annotated (e.g. the standard library). Using an example to illustrate, explain how this works.

Consider the following program written in Java:

```

1  class BoundedList {
2      private Object @NonNull [] items;
3      private int length;
4
5      public BoundedList(int size) { items = new Object[size]; }
6
7      public void add(@NonNull Object x) {
8          if(length < items.length) { items[length++] = x; }
9      }
10     public @NonNull Object get(int i) {
11         if (i < length) { return items[i]; }
12         throw new ArrayIndexOutOfBoundsException();
13     } }

```

- vi. **(2 marks)** The above program fails non-null checking. Briefly, identify what the problem is.

- vii. **(4 marks)** Briefly, discuss why the above program presents an inherent challenge for a non-null analysis.

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.

4. Design Validation

(30 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 set(int[] items, int i, int val) -> (int[] r)
requires 0 <= i && i < |items|:
//...
```

Meets precondition:

Does not meet precondition:

ii. **(2 marks)**

```
function indexOf(int[] items, int val) -> (int r)
requires some { i in 0..|items| | items[i] == val }:
//...
```

Meets precondition:

Does not meet precondition:

iii. **(2 marks)**

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

Meets precondition:

Does not meet precondition:

iv. **(2 marks)**

```
function find(int[] xs, int val) -> (int r)
requires |xs| > 1 ==> all { i in 1..|xs| | xs[i-1] < xs[i] }:
//...
```

Meets precondition:

Does not meet precondition:

(Question 4 continued on next page)

(Question 4 continued)

(b) Consider the following implementation for the function `fill()`.

```
function fill(int[] items, int item, int index) -> (int[] r):  
  //  
  if index == |items|:  
    return items  
  else:  
    items[index] = item  
    return fill(items, item, index+1)
```

i. (2 marks) Describe in your own words what the function `fill()` does.

ii. (5 marks) Provide an appropriate specification for function `fill()`.

(Question 4 continued on next page)

(Question 4 continued)

(c) Consider the following implementation for the function `cut ()`:

```

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

```

i. (2 marks) Briefly, describe in your own words what function `cut ()` does.

ii. (4 marks) Provide an appropriate *loop invariant* for function `cut ()`.

(Question 4 continued on next page)

(Question 4 continued)

(d) Consider the following implementation for the function `cmp()`:

```
1 function cmp(int[] left, int[] right, int i) -> (int r):
2     if i == |left|:
3         return 0
4     else if left[i] < right[i]:
5         return -1
6     else if left[i] > right[i]:
7         return 1
8     else:
9         return cmp(left, right, i+1)
```

i. (2 marks) Briefly, describe in your own words what function `cmp()` does.

ii. (7 marks) Provide an appropriate specification for function `cmp()`.

Student 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.

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.