Surname: . . . . . **First Name:** . . . . . **Student ID:** . . . . . .

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VUW VICTORIA UNIVERSITY OF WELLINGTON

TESTS – 2021

TRIMESTER 1

SWEN 326

SAFETY-CRITICAL SYSTEMS

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 railway*:

> "Model trains and sets of points (track switches) are operated by a *software controller*. Track sections contain *sensors* which report the number of trains within that section. Trains travel in a pattern specified by the controller, at the maximum *speed* for the current section of track and proceeding to the next track section determined by the *direction* of travel and the *points configuration*. Under no circumstances should the entire model railway system stop operating."

i. **(2 marks)** Using the terminology of IEC 61508, identify the *Equipment Under Control* for the model railway system.

The trains and the points.

ii. **(2 marks)** Using the terminology of IEC 61508, identify the part of the system on which the *Functional Safety* of the whole system relies.

The sensors.

iii. **(2 marks)** Identify a *hazard* present in the model railway system.

Both *other trains* and *sharply curved track* were identified in lectures as hazards. Either one would be an acceptable answer, but there could be other reasonable hazards identified as an answer. For example, changing the points poses a hazard.

iv. **(4 marks)** A *risk* can be described by a circumstance involving a hazard with an undesired outcome. Using this pattern, briefly describe a risk involving the above hazard.

The risks "taking a curve at high speed causes derailment and a halt to operation" and "two model trains travelling on same track collide and become damaged" were discussed in lectures, but there are other reasonable risks which could be identified. For example, "changing the points while a model train is passing through them immobilises or derails the train".

v. **(2 marks)** Briefly, discuss the ways that the hazards in a system can be *identified* when performing a hazard and risk analysis.

There are three main ways hazards can be identified: (i) the expertise and experience of others, (ii) by trials, experiments and simulations, and (iii) theoretical analysis, thinking and reasoning.

vi. **(4 marks)** Under IEC 61508 there can never be *zero risk*. Briefly, discuss what this means with respect to the model railway system.

This means that, despite whatever risk mitigation systems are in place, there is always a chance they will fail. This is referred to as the *residual risk*. For example, a software failure or a failure of the points themselves could cause the set of points entering a configuration inconsistent with that expected by the controller, resulting in collision or derailment. Alternately, a location sensor could malfunction or fail resulting in the railway system entering a deadlock situation, or two trains entering the same section of track and colliding. Furthermore, unaccounted variations between model train or tracks could lead to the actual safe maximum speed being lower than that we have calculated causing a derailment. We could add additional sensors or be more conservative in our safety margins to further mitigate this, but still there is a chance that the sensors malfunction or fail, or that too conservative a safety margin causes the railway to enter a deadlock and stop operating.

(b) Safety-critical systems often require results be computed in real-time to mitigate the risk of undesired undesired consequences:

i. **(2 marks)** Briefly describe the constraints of *Firm* and *Soft* real-time systems.

In a *Firm* real-time system, infrequent missed deadlines are tolerable but may degrade performance; results returned after a deadline have no value. In a *Soft* real-time system, the value of a result degrades after deadline, and may lead to degraded performance.

ii. **(2 marks)** Worst-Case Execution Time (WCET) is important for *Hard* real-time systems. Briefly identify *two* factors which might be needed to compute a WCET.

Four factors covered in lectures were: (i) the sequence of machine instructions to measure; (ii) the cycle time for individual machine instructions; hardware characteristics such as memory access times, sensor response times and similar and (iv) knowledge that the computation is bounded (which can be hard to prove!).

iii. **(2 marks)** A safety-critical Real-Time Operating System will have characteristics necessary to meet safety requirements. Briefly identify two such characteristics.

Five characteristics were covered in lectures: (i) determinism which provides necessary predictability, (ii) responsiveness within guaranteed timeframes, (iii) controllability by the software developer and integrator, (iv) reliability that normal operation will continue indefinitely, and (v) fail-safe so that failure does not create a hazard.

(c) Memory management is important in safety-critical systems:

i. **(2 marks)** Many microcontroller architectures provide a relatively limited data memory space. Briefly describe a risk posed by the *Stack* used for temporary data storage.

The predominant risk is that program execution causes the temporary data stack at the top of memory to become so large that it overwrites data stored in the lower part of memory, causing safety-critical system failure. This is known as a "stack overflow".

ii. **(2 marks)** In systems where memory is managed by an operating system, memory fragmentation is a hazard. Briefly describe the risk posed by memory fragmentation.

Memory fragmentation can cause an out-of-memory situation, and consequently failure of the system, when a *contiguous* block of memory is requested but is unavailable due to the available fragments being smaller than requested — even if sum of available memory is more than sufficient.

iii. **(4 marks)** JSR302 states "One of the design goals of the Safety Critical Java (SCJ) specification has been to enable the development of SCJ applications that are not vulnerable to reliability failures due to memory fragmentation." Briefly, discuss the context of this SCJ design goal.

SPARE PAGE FOR EXTRA ANSWERS

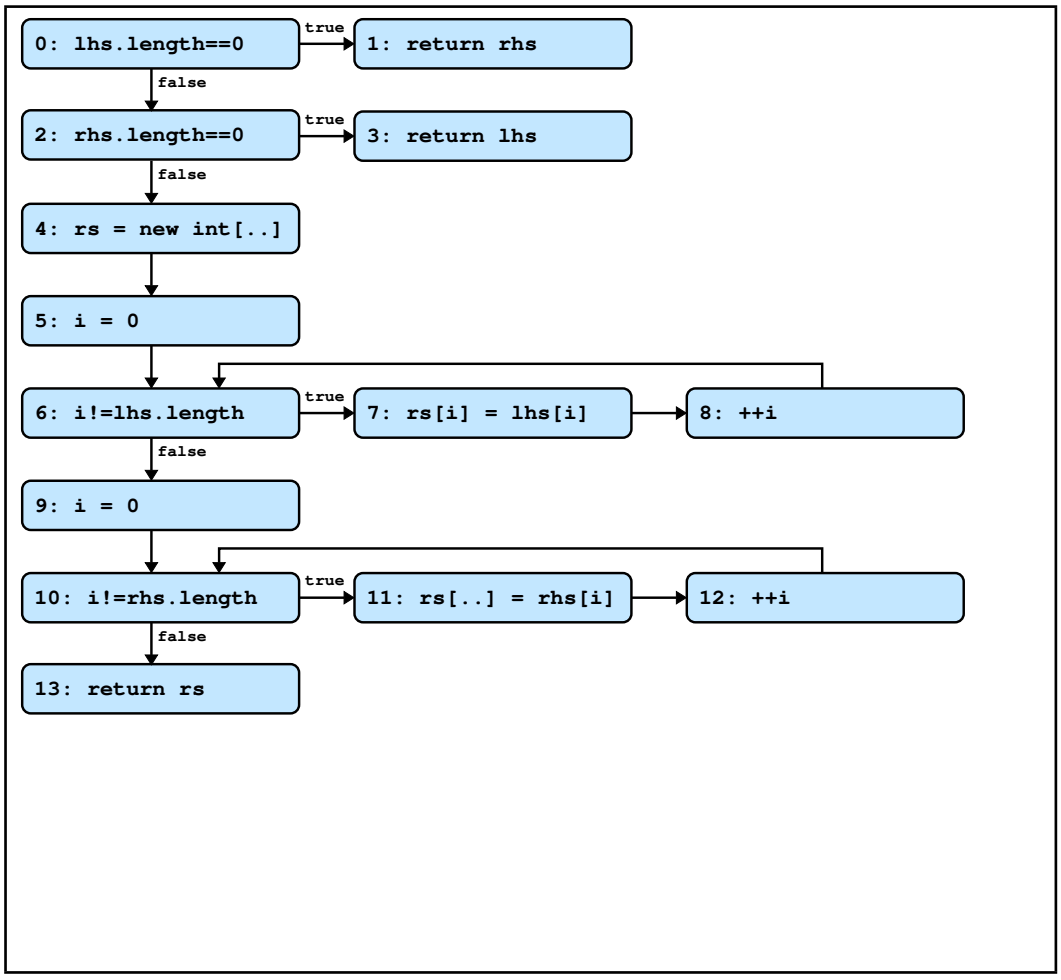
Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

2. Testing **(30 marks)**

Consider the following Java class which compiles without error:

```
1     public int[] append(int[] lhs, int[] rhs) {
2        if(lhs.length == 0) { return rhs; }
3        else if(rhs.length == 0) { return lhs; }
4        else {
5           int[] rs = new int[lhs.length + rhs.length];
6           // Copy left half
7           for(int i=0;i!=lhs.length;++i) {
8              rs[i] = lhs[i];
9           }
10          // Copy right half
11          for(int i=0;i!=rhs.length;++i) {
12             rs[i + lhs.length] = rhs[i];
13          }
14          // Done
15          return rs;
16       }
17    }
```

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



(Question 2 continued on next page)

(Question 2 continued)

(b) **(2 marks)** Briefly, discuss why the following should not be considered a valid test.

```
1    @Test public void test_0() {
2       int[] l1 = append(null,null);
3       assertTrue(l1 == null);
4    }
```

The `append()` function has an implicit precondition that both input arrays are non-null.

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

```
1    @Test public void test_1() {
2       int[] l1 = new int[]{1,2,3};
3       int[] l2 = new int[0];
4       int[] l3 = append(l1,l2);
5       assertTrue(l3 == l1);
6    }
```

```
1    @Test public void test_2() {
2       int[] l1 = new int[0];
3       int[] l2 = new int[]{1,2,3};
4       int[] l3 = append(l1,l2);
5       assertTrue(l3 == l2);
6    }
```

```
1    @Test public void test_3() {
2       int[] l1 = new int[]{1,2,3};
3       int[] l2 = new int[]{2,3,4};
4       int[] l3 = append(l1,l2);
5       assertTrue(Arrays.equals(l3,new int[]{1,2,3,2,3,4}));
6    }
```

(Question 2 continued on next page)

(Question 2 continued)

(d) **(3 marks)** Briefly, discuss whether 100% *Modified Condition/Decision Coverage* (MC/DC) is achievable for the `append()` method.

Modified Condition / Decision Coverage is concerned with ensuring each condition independently affects it enclosing decision. However, in the `append()` method, every decision is a single condition. Hence, MC/DC is essentially the same as branch coverage for this example.

(e) *Unrealisable paths* are a significant challenge for computing code coverage.

i. **(2 marks)** Briefly, state what an unrealisable path is.

An unrealisable path is a valid path through the control-flow graph which cannot beexecuted in practise due to specific logical conditions on the path being in contradiction.

ii. **(2 marks)** Identify an unrealisable path in the `append()` method.

The path which passes through the CFG reaching the final **`return`** statement, but without entering either of the loops is unrealisable. That is, the path $0 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 13$.

(f) An important test critereon is *simple path coverage*.

i. **(2 marks)** Briefly, state what simple path coverage is.

The proportion of *simple paths* covered, where a simple path is a path through the CFG where no two nodes are identical, other than the first and last.

ii. **(3 marks)** Briefly, discuss whether 100% *simple path coverage* is achievable for the `append()` method.

Since there is an unrealisable simple path (as highlighted in e(ii) above), it is impossible to obtain 100% simple path coverage.

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) Computing the *Worst-Case Execution Time (WCET)* for a given instruction sequence is important for real-time embedded systems.

For this question, you may assume it takes exactly *one cycle* to execute a single instruction.

```
0x0000:  rjmp 5
0x0001:  cp r24, r22
0x0002:  cpc r25, r23
0x0003:  brge 1
0x0004:  movw r16, r12
0x0005:  ret
0x0006:  push r28
0x0007:  push r29
0x0008:  movw r24, r8
0x0009:  rcall -9
0x000A:  mov r22, r28
0x000B:  pop r29
0x000C:  pop r28
```

i. **(2 marks)** Determine the WCET for the above instruction sequence (in cycles).

13 cycles

ii. **(5 marks)** Briefly, outline how a *control flow analysis* can be used for computing WCET. You should use the instruction sequence shown above to illustrate.

A control-flow analysis is a form of static analysis which explores all possible execution paths in the program without executing it. Furthermore, it is a lightweight analysis which does not attempt to track information about the contents of registers or other memory locations. When a branch is encountered (such as `brge` in the example above), the analysis will explore *both* paths. The worst-case execution time will then the time taken to execute the *longest path* in the Control-Flow Graph.

iii. **(5 marks)** Static analyses are often said to be *conservative*. Briefly, discuss what this means in the context of WCET.

Since static analyses typically do not track information about the contents of registers or memory locations, they must make conservative choices when e.g. they encounter conditional branches. To resolve this, they will always explore both possible outcomes, even in situations where only one outcome is possible (e.g. perhaps whether the other outcome is part of some unrealisable path). Thus, the computed WCET may be an *over approximation* of the actual WCET.

iv. **(4 marks)** Briefly, discuss why *loops* present a significant challenge when computing WCET using control-flow analysis.

Loops present a significant challenge for a static analysis such as this. In particular, such analyses do not have knowledge about the number of iterations a loop will iterate. This means they have to make conservative choices (as above) when loops are encountered. Unfortunately, in this case, the only suitable choice would be to set the WCET as *infinite* which is not a particularly informative result.

v. **(4 marks)** Briefly, discuss why *indirect jumps* present another challenge when computing WCET using control-flow analysis.

Indirect jumps present similar problems, as they branch to an address based on the contents of a register. If the analysis has no knowledge of the possible values for this register, then *it will be unable to proceed*. Again, this would mean reporting an infinite WCET. Some techniques, such as integer range analysis, can be used here to give the analysis some knowledge of the possible targets for an indirect jump.

(b) An alternative way of computing the WCET for an instruction sequence is to use *explicit-state model checking*.

i. **(4 marks)** Briefly, outline how explicit-state model checking *differs* from using a control-flow analysis.

The control-flow analysis considered is a lightweight static analysis which records only a minimal amount of information about the contents of registers and memory locations and, as such, can face insurmountable problems with loops and indirect jumps. In contrast, explicit-state model checking records a significant amount of information about the machine state at each point during the analysis. For example, it may have concrete values for some or all of the available registers or memory locations. Likewise, when a fork arises due to e.g. a conditional branch, the model checker may be able to concretise unknown values on either branch.

ii. **(6 marks)** Briefly, discuss the *pros* and *cons* of using explicit-state model checking for calculating WCET.

The main **benefits** of explicit static model checking arise from a significantly improved knowledge about the contents of registers and memory locations. This can offer most precise estimates of the WCET for a given program. In particular, the model checker may be able execute bounded loops and/or indirect branching without any trouble.

The main **downsides** of explicit state model checking are that it requires significantly more resources (i.e. memory and execution time). In some cases, it may not be able to analyse the program in a reasonable amount of time or in a reasonable amount of space. Considerable effort is required to ensure good compression of states at each point, and its likely that manual tuning is required for larger programs.

SPARE PAGE FOR EXTRA ANSWERS

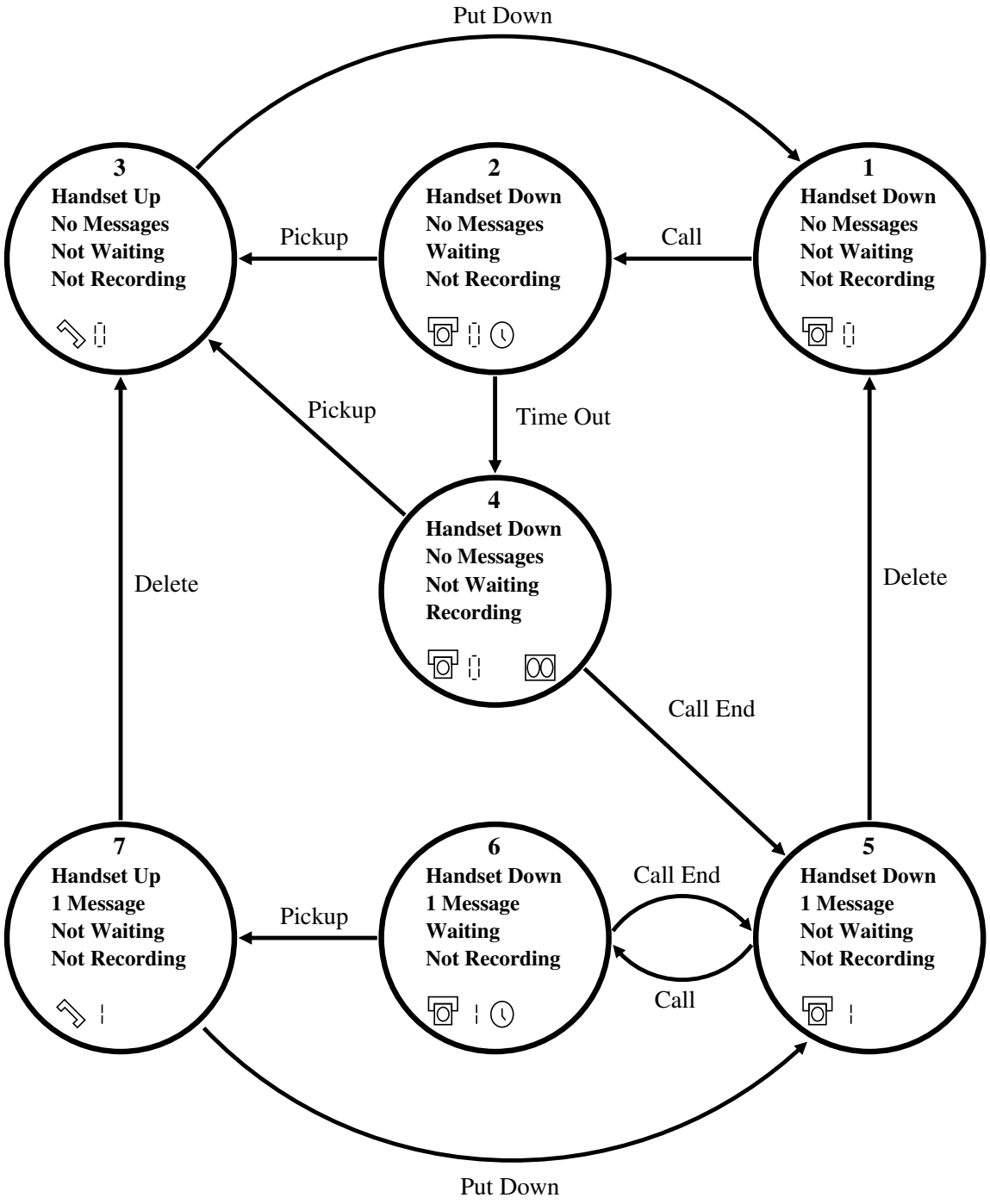
Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

4. Design Validation **(30 marks)**

Consider the following description of a simple *answering machine*:

> "The answering machine can store up to one phone message, and has a handset for answering calls manually. When an incoming call rings, a timer is started. If the user picks up the handset before the timer elapses, then he/she may answer the call. Otherwise, the machine begins to record the message. At this point, the user may still pick up the handset and answer the call. "

A *state machine diagram* for the answering machine has been provided:



(a) For each of the following statements, indicate whether you think it is a true or false statement based on the state machine diagram.

i. **(2 marks)** If the handset is down and an incoming call occurs, the machine immediately begins recording.

False

ii. **(2 marks)** If the machine is recording a message, the user can always pick up the phone and answer it.

True

iii. **(2 marks)** If an incoming call occurs and the handset is up, the machine will record a message.

False

iv. **(2 marks)** The user may delete a message whilst answering a call.

True

v. **(2 marks)** If the machine has already stored a message, then it will not record another until the first is deleted.

True

(b) Provide a suitable *execution trace* for the following scenarios. For each state, you need only list its number. Your execution trace may start from any state you chose.

i. **(2 marks)**

> *"When the phone rang, Jane answered it immediately. It was a wrong number, so she put the handset down."*

$1 \rightarrow 2 \rightarrow 3 \rightarrow 1$

ii. **(2 marks)**

> *"John deleted the message. Then, the phone rang, but he could not answer it. So, a new message was recorded."*

$5 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5$

(c) Consider the following *incomplete* Whiley model of the answer machine:

```
1  type State is {
2     bool handsetUp,
3     bool waiting,
4     bool recording,
5     int messages
6  }
7  where messages >= 0
8  // Cannot be waiting and recording
9  where !(waiting && recording)
10
11 function call(State s1) -> (State s2)
12 // handset cannot be up
13 requires !s1.handsetUp
14 // cannot be already waiting
15 requires !s1.waiting
16 // cannot be recording
17 requires !s1.recording:
18    s1.waiting = true
19    return s1
20
21 function callEnd(State s1) -> (State s2)
22 // must be either recording or waiting
23 requires s1.recording || s1.waiting:
24    if s1.recording:
25       s1.messages = s1.messages + 1
26    s1.recording = false
27    s1.waiting = false
28    return s1
29
30 function pickup(State s1) -> (State s2)
31 // handset must be down
32 requires !s1.handsetUp
33 // must be waiting
34 requires s1.waiting:
35    s1.handsetUp = true
36    s1.waiting = false
37    return s1
```

i. **(2 marks)** Give a valid instance of `State` which corresponds to state four from the diagram on page 14.

```
{handsetUp: false, waiting: false, recording: true, messages: 0}
```

ii. **(2 marks)** Give a valid instance of `State` which does not correspond to *any* state from the diagram on page 14.

```
{handsetUp: false, waiting: false, recording: false, messages: 2}
```

iii. **(4 marks)** Complete the following function to model the "Delete" transition from the diagram on page 14.

```
1 function delete(State s1) -> (State s2)
  // must have a recorded message
  requires s1.messages > 0
  // cannot be waiting or recording
  requires !(s1.waiting || s.recording):
     s1.messages = s1.messages - 1
```

iv. **(2 marks)** The function `callEnd()` allows more transitions than described in the state machine diagram on page 14. Briefly, explain why.

The `callEnd()` transition can occur whilst the machine is *waiting* but not yet *recording* (i.e. transition $2 \rightarrow 1$ is possible in the Whiley model, but not the state machine diagram).

v. **(2 marks)** What key functionality is provided by the Whiley model but not the state machine diagram on page 14?

The Whiley model allows the machine to record more than one message.

(d) **(4 marks)** Briefly, discuss the *pros* and *cons* of using formal languages (such as Whiley) compared with hand-drawn diagrams (such as that on page 14)

Formal languages like Whiley can be **checked** for correctness using verification tools and similar. They are also unambiguous in their meaning, and thus offer a clear description of how the machine operates. They also make it easy to encode more complexity into the model (for example, an arbitrary number of messages was easy to specify in Whiley). On the downside, formal languages can be difficult to understand and visualise compared with the diagram. They also require programming expertise.

* * * * * * * * * * * * * * *

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.

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.