This is a CLOSED-BOOK quiz.

Before you start, write your name at the top of every sheet.

There are 5 parts, labeled A through E. Please check your copy of the quiz before you start to make sure it is complete: you should have 6 sheets, printed on 10 sides in total. Throughout the quiz, state machines have the "blocking" interpretation that we have assumed all the time, in which events that are not shown explicitly by a machine are blocked rather than ignored.

You have 75 minutes and should attempt to answer all questions. Note that part C is worth more than the other parts. Good luck!

<table>
<thead>
<tr>
<th>Part</th>
<th>Score</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Name:
A State Machine Basics

True or false?

1. T F All state machines are finite.

2. T F A state machine has at most one initial state.

3. T F A state machine has at least one initial state.

4. T F A state machine transition cannot start and end at the same state.

5. T F An invariant must hold in the initial state.

6. T F An invariant must hold in every reachable state.

7. T F An invariant must hold in every state.

8. T F State machine events are assumed to be instantaneous.

9. T F States are assumed to be occupied only instantaneously.

10. T F In state machine testing, path coverage implies transition coverage.

11. T F In state machine testing, transition coverage implies state coverage.

12. T F In state machine testing, state coverage implies transition coverage.

13. T F Path coverage is often infeasible.

14. T F The Object as Machine Pattern associates events with methods.

15. T F The Singleton Pattern associates events with static methods.

16. T F The State Pattern associates events with objects.

17. T F A liability of the Object as Machine pattern is that state components must be passed around.

18. T F In a state machine with parallel submachines, two distinct events can occur simultaneously.
B State Machine Modelling

A car’s cruise control system is activated by the driver setting the speed, and is deactivated when the driver applies the brake. It can be reactivated by pressing the resume button (which causes the car to return to the last set speed), or by setting the speed again.

The diagram shows a skeleton of a state machine model, with labels missing.

Add labels choosing from the following list:

- speed < setSpeed
- speed > setSpeed
- resume
- set
- brake
- BRAKING
- RESUMING
- SETTING
- ON
- OFF
**C Concurrency and Invariants**

Alice and Ben share a fridge and attempt to keep it stocked with milk, but are eager to avoid having more than one carton of milk in the fridge at a time.

The state machine diagram below models the behaviours that the fridge allows. The event labels and their interpretations are:

- **aobs**: Alice observes that the fridge is empty
- **bobs**: Ben observes that the fridge is empty
- **abuy**: Alice places in the fridge a new carton of milk that she has bought
- **bbuy**: Ben places in the fridge a new carton of milk that she has bought
- **adrink**: Alice drinks some of the milk in the fridge, reducing the supply
- **bdrink**: Ben drinks some of the milk in the fridge, reducing the supply

1. Which of the following is a trace of the state machine as shown? Circle Y (yes) or N (no) for each.
   a. Y N < >
   b. Y N < adrink >
   c. Y N < aobs, bobs >
   d. Y N < abuy, aobs >
   e. Y N < abuy, abuy >
   f. Y N < abuy, bdrink, bdrink >
   g. Y N < abuy, bdrink, bobs >
   h. Y N < abuy, abuy, abuy >
Now consider a fuller model with three submachines running in parallel: the fridge machine from before, and machines representing each of Alice and Ben. The machine representing Alice, for example, has states $\text{AIN}$ and $\text{AOUT}$ representing whether Alice is in the apartment drinking milk or outside obtaining it.

2. True or false?
   a. T F Adding the machines for Alice and Ben to the fridge machine reduces the trace set
   b. T F In such a model, every event must be participated in by all parallel submachines
   c. T F In such a model, a submachine can always take a step by itself

3. The number of states in the product machine in this case is:
   a. 6
   b. 8
   c. 16

4. Which of the following traces is a trace of the new machine? Circle Y (yes) or N (no) for each.
   a. Y N < adrink >
   b. Y N < aobs, bobs >
   c. Y N < abuy, aobs >
   d. Y N < aobs, abuy >
   e. Y N < aobs, abuy, abuy>
   f. Y N < aobs, bobs, abuy, bbuy>
5. True or false? (where ⇒ means logical implication)
   a.  T  F  (AOUT ⇒ EMPTY) is an invariant
   b.  T  F  (AOUT or BOUT ⇒ EMPTY) is an invariant
   c.  T  F  (EMPTY ⇒ AOUT or BOUT) is an invariant

Because this scheme sometimes results in too much milk being bought, Alice and Ben decide to alternate in their purchasing. It happens also that Alice is in fact more generous; when she buys milk, she buys a carton twice the size of the carton that Ben buys. The new scheme can be modelled textually like this:

```
state
  turn: {Alice, Ben}
  milk: int

init
  turn = Alice and milk = 0

op drink (p: {Alice, Ben})
  when milk > 0
  do milk = milk - 1

op buy (p: {Alice, Ben})
  when turn = p and milk = 0
  do
    if p = Alice
      then milk = 20; turn = Ben
    else milk = 10; turn = Alice
```

6. True or false?
   a.  T  F  (milk ≤ 20) is an invariant
   b.  T  F  (milk > 0) is an invariant
   c.  T  F  (turn = Alice ⇒ milk ≤ 10) is an inductive invariant
   d.  T  F  (turn = Alice ⇒ milk ≤ 10) is an invariant
   e.  T  F  (eventually milk = 0) is an invariant

7. The property (turn = Ben ⇒ milk ≤ 20) is an invariant. Establishing that buy preserves this invariant requires considering
   a.  Y  N  The initialization
   b.  Y  N  The operation’s firing condition (“when”)
   c.  Y  N  The operation’s postcondition (“do”)
Alice and Ben find this scheme unsatisfactory, because if one of them goes on vacation, the other one remains thirsty. So they try yet another scheme, in which the person buying the milk must first leave a note, and then remove it on returning with the milk. This is shown in the state machine below, with the event aleave corresponding to Alice noticing there is no milk and leaving a note, and aremove corresponding to Alice removing the note.

8. True or false?
   a. T F This scheme successfully eliminates traces containing abuy immediately followed by bbuy
   b. T F One purpose of adding the note submachine is to prevent deadlock
   c. T F The addition of the note submachine establishes the invariant (not (AOUT and BOUT))
   d. T F The note submachine acts as a condition variable
   e. T F The note submachine acts as a lock
   f. T F The note submachine establishes mutual exclusion
D  State Machine Patterns

1. True or false?
   a. T F  If avoiding runtime allocation is important, the Singleton pattern should be considered
   b. T F  The Object as Machine pattern can be used without allocating objects on transitions
   c. T F  In its simplest form, the State Pattern allocates an object on a state transition

2. True or false?
   a. T F  The Singleton pattern is good for a machine that will be used as a plugin
   b. T F  Object as Machine is the most common state machine pattern
   c. T F  The State pattern is most commonly used in embedded systems

3. True or false?
   a. T F  When states are nested (so that states have many substates, but the substates of each larger state are different), the State pattern is a good choice
   b. T F  When a state machine is composed of multiple submachines running in parallel, the State pattern is a good choice
   c. T F  When a state machine has very many distinct top level states, the State pattern is a good choice

Dynamic dispatch is the term often used to describe how the choice of method depends on the runtime type of the receiver object.

4. True or false?
   a. T F  The Singleton pattern does not use dynamic dispatch at all
   b. T F  The State pattern relies heavily on dynamic dispatch
   c. T F  The switch statement in the Switch on Enumeration pattern is not really object-oriented
E Interfaces

Ben Bitdiddle, a Java novice, is designing an application SuperAwesomeApp that uses a datatype he’s invented called SuperFastList. He looked over the lecture notes of 6.005, and realized that he should decouple the application from the datatype with an interface, so he came up with the following dependence diagram, in which SuperFastCollection is an interface. Unfortunately, he’s a bit lost and needs some advice, so he’s called you up since he knows you actually attended the lecture and can help him out.

1. True or false?
   a. T F The diagram should also show a dependence of SuperAwesomeApp on SuperFastList, since a class cannot depend on an interface alone, and must have access to the implementation
   b. T F SuperFastList depends on SuperFastCollection because it implements it, and a change to the interface will likely imply a change to the class
   c. T F If this diagram is correct, SuperFastList cannot be instantiated inside SuperAwesomeApp, but an object of type SuperFastList can be passed to SuperAwesomeApp

Ben tells that you that his friend Alyssa P. Hacker pointed out that SuperFastList correctly implements all the methods declared in the java.util.List interface. She suggested to him that he should replace the interface SuperFastCollection in the diagram by java.util.List. Ben wants your opinion.
2. True or false?
   a. T F Replacing SuperFastCollection by java.util.List would be desirable if possible
   b. T F Replacing SuperFastCollection by java.util.List will not be possible if SuperFastCollection contains additional methods that are not in java.util.List
   c. T F Replacing SuperFastCollection by java.util.List may not be possible, because SuperAwesomeApp may call methods that do not belong to java.util.List
   d. T F Replacing SuperFastCollection by java.util.List may not be possible, because SuperFastList may contain methods that do not belong to java.util.List

Ben discovers that his application needs an extra list method conquerWorld that is not currently a method of SuperFastList. So he creates a subclass ExtraSuperFastList of SuperFastList. Currently, SuperAwesomeApp uses only one superfast list that is created outside and passed in as a parameter. Alyssa warns Ben that he may need to change how this parameter is passed.

3. In order for SuperAwesomeApp to call this new method, what should the declared type / runtime type of the parameter be? Circle one answer.
   a. SuperFastCollection / SuperFastList
   b. SuperFastCollection / ExtraSuperFastList
   c. SuperFastList / ExtraSuperFastList
   d. ExtraSuperFastList / ExtraSuperFastList
   e. none of the above

4. Suppose classes A and B implement the interface C, and that A, B and C have sets of methods MA, MB and MC. Which of the following constraints always holds? Circle yes (Y) or no (N) for each.
   a. Y N MA ⊆ MC
   b. Y N MC ⊆ MA
   c. Y N MA ∪ MB = MC
   d. Y N MA ∩ MB ⊆ MC

5. Which of the following are allowed in an interface declaration? Circle yes (Y) or no (N) for each.
   a. Y N extension of multiple interfaces
   b. Y N static method signatures
   c. Y N non-static method signatures
   d. Y N method bodies
   e. Y N constant definitions
   f. Y N field declarations