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

EXAMINATIONS — 2009

END-OF-YEAR

SWEN 102
Introduction to Software Modelling

Time Allowed: 3 Hours

Instructions: There are 180 possible marks on the exam.
Answer all questions in the boxes provided.
Every box requires an answer.
If additional space is required you may use a separate answer booklet.
Some example Alloy code is provided on the last page.
Non-electronic foreign language dictionaries are allowed.
Calculators ARE NOT ALLOWED.
No other reference material is allowed.

Question	Topic	Marks
1.	Use Case Diagrams	30
2.	Object and Class Diagrams I	30
3.	Object and Class Diagrams II	30
4.	Writing Invariants	30
5.	Using Alloy	30
6.	State Machines	30
	Total	180

Student ID: . . . .

Question 1. Use Case Diagrams [30 marks]

(a) [3 marks] Perform a *textual analysis* on the following description, to find candidate use cases.

You should carefully and neatly underline key verb phrases in the text in the box.

> SourceForge is a website used for software development. Software developers develop programs and upload new versions. Users search the website looking for programs to download and use.
>
> To upload a new version of a program, the developer provides a log message, and version number. The log message details what changes were made. Version numbers must always increase when new versions are uploaded.
>
> The creator of a project can edit the information shown about their projects. This includes the title, a short description and which programming languages are used. They can also set permissions allowing other users to upload new versions to the project.
>
> Users search the website for software based on keywords they supply. Users can search the website without needing to create an account on SourceForge. Developers must create an account before they can create or work on a project. They must log in whenever they want to work on a project.

(Question 1 continued on next page)

Student ID: . . . .

(Question 1 continued)

(b) [12 marks] Draw **essential use case cards** for the following **three** use cases in this system.

Find project by searching

Upload new version of program

Change project information

(Question 1 continued on next page)

Student ID: . . . .

(Question 1 continued)

(c) [9 marks] Draw a **use case diagram** showing at least 3 actors and at least 6 use cases that you would produce in a model of this system.

(Question 1 continued on next page)

Student ID: . . . .

(Question 1 continued)

(d) [6 marks] Give characteristics for two actors in the system.

1. **Actor Name**
2. Domain Knowledge
3. System Knowledge
4. Frequency of Interaction

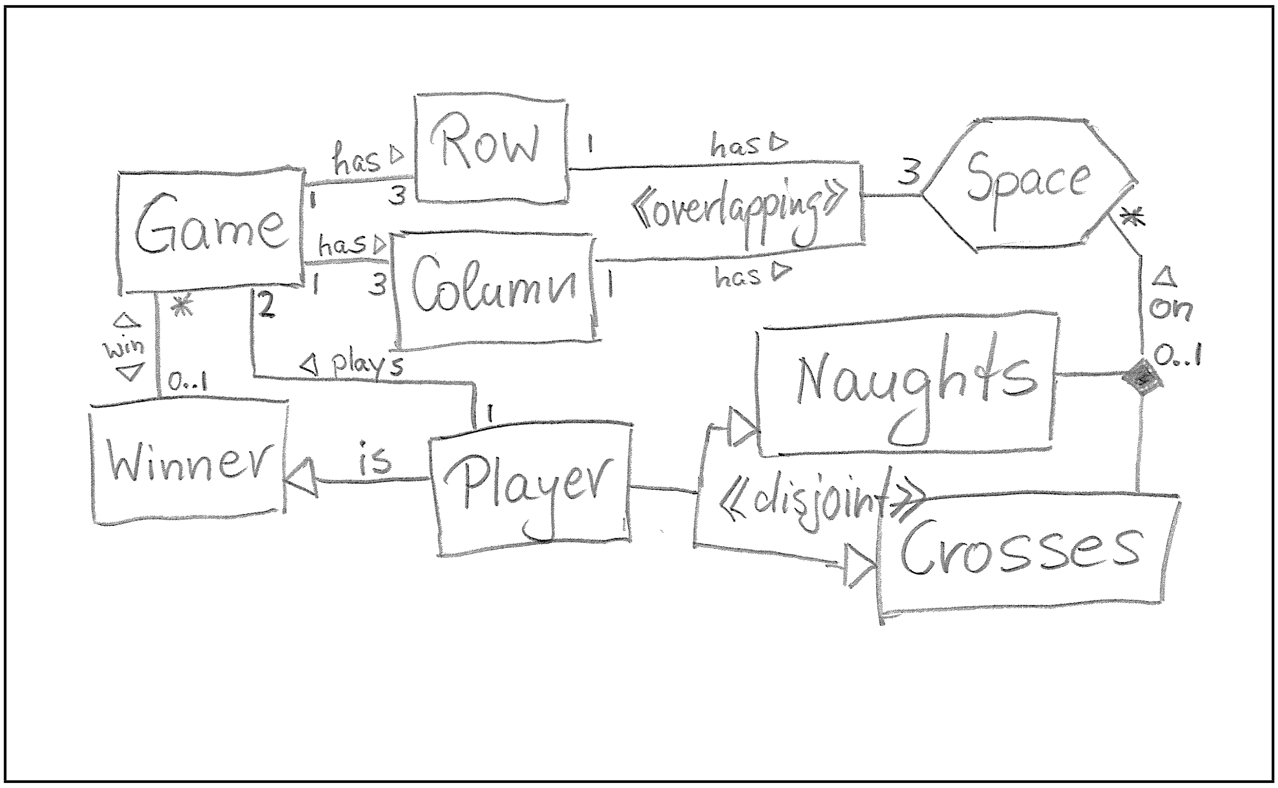
1. **Actor Name**
2. Domain Knowledge
3. System Knowledge
4. Frequency of Interaction

Student ID: . . . .

Question 2. Object and Class Diagrams I [30 marks]

Your are planning to implement the naughts and crosses game. The class diagram below shows an attempt to model the system according to the description given next.

> Naughts and crosses is a game for two players who take turns marking the spaces in a grid. First, the players agree on who is playing Naughts and who is playing Crosses. A game grid consists of 3 rows and 3 columns. A space is where a row and a column overlap. Spaces are initially empty and then get occupied by either Naughts or Crosses, but not by both. The winner is the player who is the first in owning a row, column, or diagonal. However, some games finish without a winner.



(Question 2 continued on next page)

Student ID: . . . .

(Question 2 continued)

(a) [15 marks] Circle and number **five distinct** problems in the class diagram on the previous page. Describe why each problem is a problem and how you would solve it.

1.

2.

3.

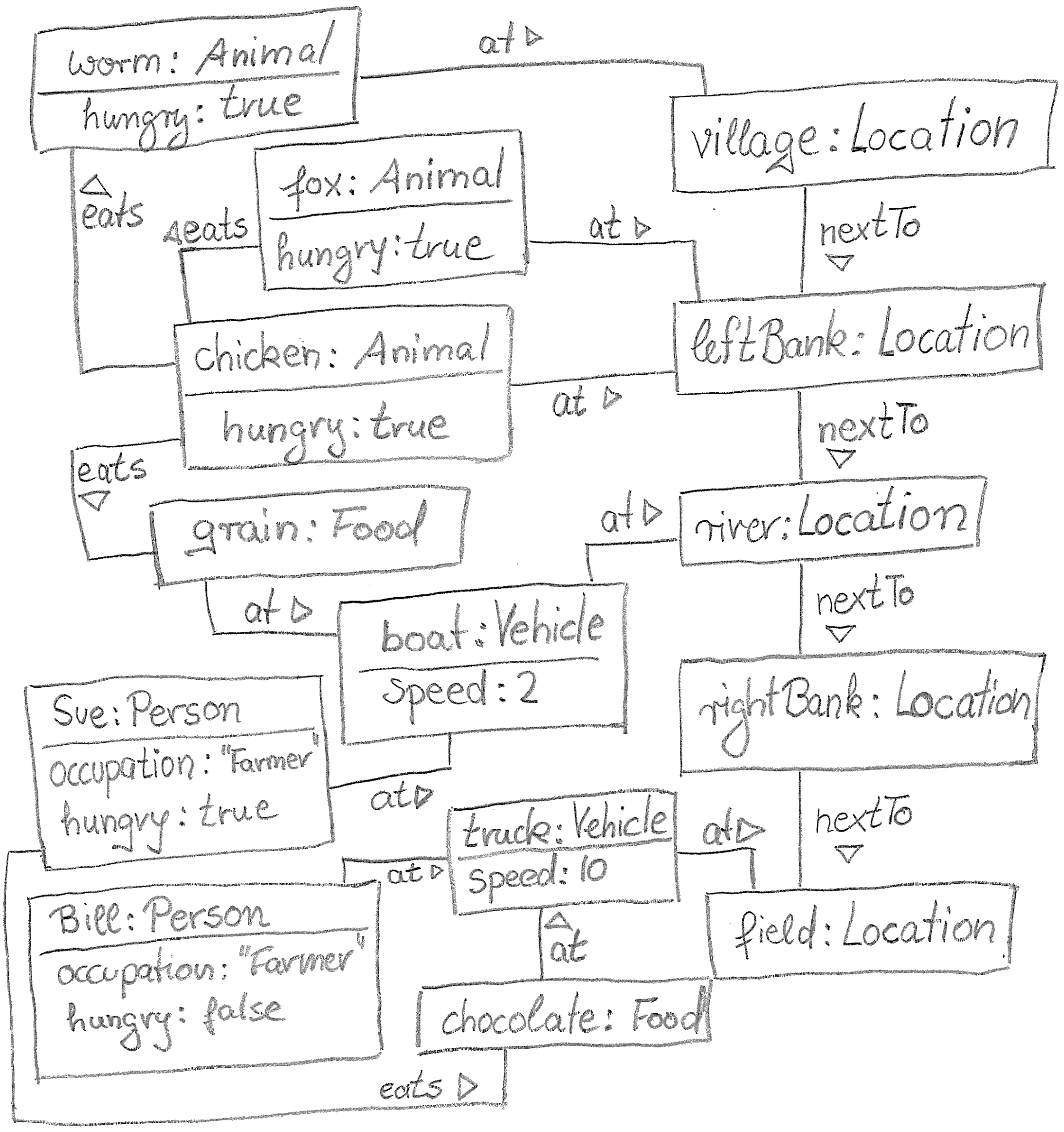
4.

5.

(Question 2 continued on next page)

(Question 2 continued)

(b) [15 marks] Consider the object diagram below and draw a corresponding class diagram on the facing page.



(Question 2 continued on next page)

Student ID: . . . .

(Question 2 continued)

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.

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.

continued...

Student ID: . . . .

Question 3. Object and Class Diagrams II [30 marks]

Consider the following text, which describes household items commonly found in a kitchen.

> There are many items of different sizes that can be found in a kitchen. Plastic bowls are useful for storing things in the fridge and for mixing ingredients. Saucepans are used to cook food or hold smaller items. When not being used, saucepans are often stored in a cupboard and may be placed inside other saucepans to save space. Plates are important! There are different kinds of plate, including side plates, main plates and serving plates. Bowls are similar to plates. They can be used to hold fruit and small bowls can be stacked inside large ones. Knives and Forks are small items that can be placed inside saucepans and bowls if needed. Chopping knives are for preparing food and are stored in a knife block. They can be blunt sometimes and may need sharpening. Drinks are served in cups, such as coffee cups and tea cups. Cups are similar to bowls, except they have handles.

(a) [5 marks] From the description above, identify ten household items that are found in a kitchen, including four containers to put items in.

(Question 3 continued on next page)

Student ID: . . . .

(Question 3 continued)

(b) [15 marks] Draw a **class diagram** to model these household items.

(Question 3 continued on next page)

Student ID: . . . .

(Question 3 continued)

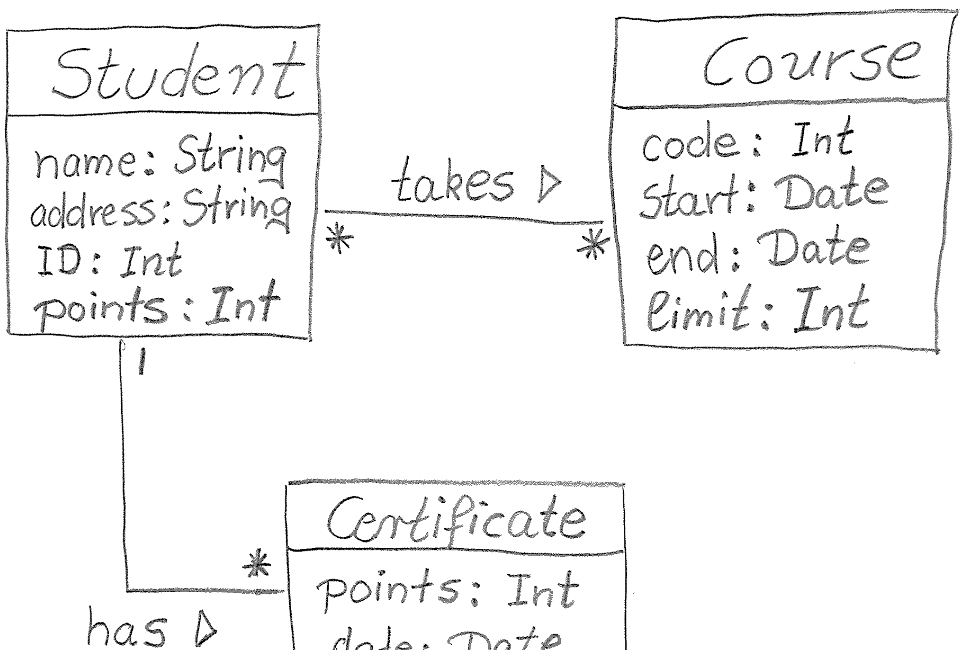
(c) [10 marks] Discuss an important aspect of the real world not modelled by your class diagram, and indicate how you would fix this. You should illustrate your discussion with diagrams where appropriate.

Student ID: ....

Question 4. Writing Invariants [30 marks]

Consider the following description of an *adult learning program*, which is made up of some *text* and a *class diagram*:

> The adult learning program records what courses students currently take and what certificates have been issued for the successful completion of a course. Students must provide their details (name and address) and are then associated with an id that uniquely identifies them.
>
> Each course has a course code, start date, end date, and a limit on the number of students that can take the course. If demand for a particular course is especially high, another course with the same course code might be offered.
>
> Students who complete a course successfully receive a certificate that includes the number of points the student received for the course. The system also records the number of points a student received in total. Students are not allowed to re-take a course for which they already have a certificate.



(Question 4 continued on next page)

Student ID: ....

(Question 4 continued)

(a) [10 marks] By considering the given text and class diagram, identify (in English) five *candidate invariants*:

(Question 4 continued on next page)

Student ID: . . . .

(Question 4 continued)

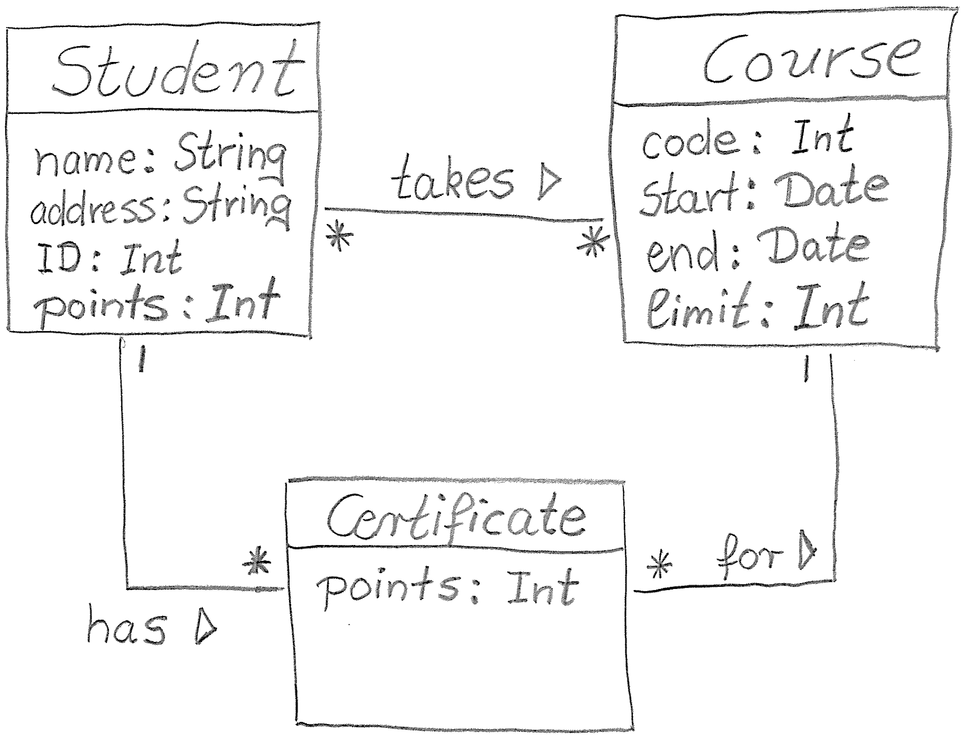
(b) [10 marks] Translate your candidate invariants from **(a)** into the Alloy-like syntax presented in lectures (example Alloy code is provided on page 33):

(Question 4 continued on next page)

Student ID: ....

(Question 4 continued)

(c) The adult learning program was changed to record the course for which a certificate was issued rather than just the course code and the date.



The following *invariants* are given for the extended system.

```
Student invariant:
  has.for in takes

Global invariants:
  all c1, c2: Course | (some c1.~for and some c2.~takes) implies
                            c1.end <= c2.start

  some d: Date | all c: Course |
        some c.~takes implies c.start <= d && d <= c.end
```

(Question 4 continued on next page)

Student ID: . . . .

(Question 4 continued)

(i) [6 marks] Translate the three invariants given in Alloy notation on the previous page into written English:

(ii) [4 marks] Are these invariants consistent? Explain why or why not.

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.

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.

Student ID: ....

Question 5. Using Alloy [30 marks]

Consider the following description of a file system:

> "A file system consists of a hierarchy of file system objects, which are either directories or files. This hierarchy has the form of a single tree with the root directory at the top containing file system objects, who in turn can contain file system objects, and so on. Files cannot contain other file system objects. Each file system object, apart from the root directory, is contained in exactly one directory called its parent. Furthermore, no cycles are allowed. That is, a file system object is never an ancestor (parent or parent of parent etc.) of itself."

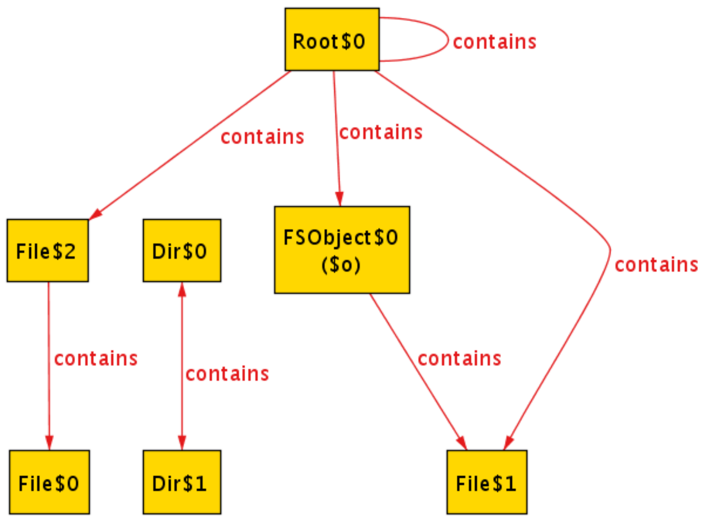
An *incorrect* model of a file system in Alloy and an *object diagram* of that model is given below:

```
// file system objects
sig FSObject {
  contains: set FSObject
}

// directories
sig Dir extends FSObject {}

// the root directory
one sig Root extends Dir {}

// files
sig File extends FSObject {}
```



The above object diagram was generated using the command "`run {} for 7`" from the Alloy model.

(Question 5 continued on next page)

(Question 5 continued)

(a) [5 marks] Evaluate each of the following Alloy expressions on the object diagram given on the previous page:

```
Root$0.contains
```

```
File$1.~contains
```

```
Dir$0.contains.contains & Dir
```

(Question 5 continued on next page)

Student ID: . . . .

(Question 5 continued)

(b) [10 marks] Circle and number five distinct ways in which the object diagram given on the previous page is *inconsistent* with the description of a file system. For each, write a brief (i.e. one line) **description of the problem** in the corresponding box below.

1)

2)

3)

4)

5)

(Question 5 continued on next page)

(Question 5 continued)

(c) [15 marks] For each problem identified in **(b)**, indicate what changes you would make to the Alloy model to fix it and **give Alloy code to illustrate**.

1)

2)

3)

4)

5)

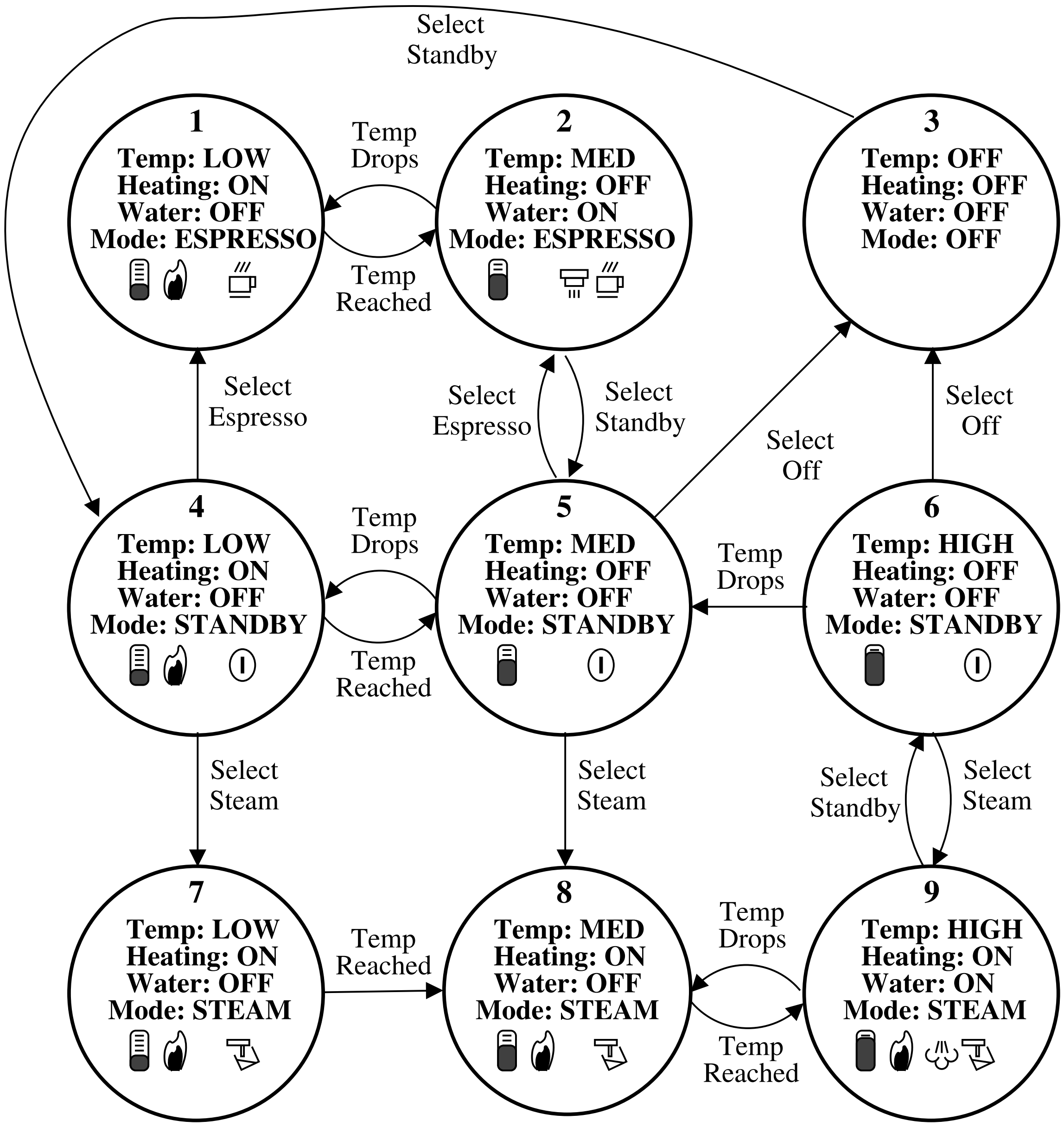
Student ID: . . . .

Question 6. State Machines [30 marks]

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

> "The Espresso Machine makes coffee by pushing hot water through ground coffee beans. A steam function is provided for steaming milk. A *control dial* is used to operate the machine. It has four positions: *Standby*, *Espresso*, *Steam* and *Off*. In *Standby mode*, the water is heated, and a *heat sensor* turns the heating off when the temperature is reached, and on again when it drops. In *Espresso mode* the machine passes hot water out into the ground coffee, whilst in *Steam mode* it shoots very hot steam out into the milk."

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



(Question 6 continued on next page)

Student ID: . . . .

(Question 6 continued)

(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] The machine continuously heats the water when "Steam" is selected.

(ii) [2 marks] The water achieves the highest temperature when "Espresso" is selected.

(iii) [2 marks] When "Espresso" is selected, the machine always pushes water out immediately.

(iv) [2 marks] After steaming milk, the machine must cool down before it can make espresso again.

(v) [2 marks] When the machine overheats, it automatically shuts down.

(b) Provide a suitable *execution trace* for the following scenarios. Your execution trace may start from whichever state you chose.

(i) [2 marks]
"John selected steam, but the machine was cold and it took a long time before he got steam."

(ii) [2 marks]
"Jane turned the machine on in the morning. When she selected Espresso later on, water came out immediately."

(Question 6 continued on next page)

(Question 6 continued)

(c) Consider the following *incomplete* Alloy model of the espresso machine:

```
abstract sig Mode {}
sig Espresso extends Mode {}
sig Steam extends Mode {}
sig StandBy extends Mode {}
sig Off extends Mode {}

sig MachineState {
 sensor : Int,
 heating : Bool,
 water : Bool,
 mode : Mode
}{
 sensor >=0 and sensor <= 3
}

pred TempReached(s1, s2 : MachineState) {
 s1.heating = True
 s2.mode = s1.mode
 s2.heating = True iff s2.mode = Steam
 s2.water = s1.water
 s2.sensor = s1.sensor + 1
}

pred TempDrops(s1, s2 : MachineState) {
 s2.mode = s1.mode
 s2.sensor = s1.sensor − 1
 s2.heating = True iff s2.mode = Steam
 s2.water = False
}

pred SelectStandBy(s1, s2 : MachineState) {
 s2.mode = StandBy and s1.mode != StandBy
 s2.heating = True iff s1.sensor <= 1
 s2.sensor = 1
 s2.water = False
}

sig ExecutionTrace { states : seq MachineState }{
 states[0].sensor = 0
 states[0].mode = Off
 states[0].water = False
 states[0].heating = False
 all disj i,j : states.inds | j=i+1 implies (TempReached[states[i],states[j]]
   or TempDrops[states[i],states[j]]  or SelectStandBy[states[i],states[j]])
}
```

(Question 6 continued on next page)

Student ID: ....

(Question 6 continued)

(i) [2 marks] Give an instance of MachineState which corresponds to state five from the diagram on page 26.

(ii) [2 marks] Give an instance of ExecutionTrace which corresponds to the execution trace "$3 \rightarrow 4 \rightarrow 5$" from the diagram on page 26.

(iii) [4 marks] Give an instance of ExecutionTrace which does not correspond to any execution trace of the diagram on page 26.

(iv) [2 marks] Can the Alloy model given ever reach a *deadlock*? Briefly justify your answer.

(Question 6 continued on next page)

(Question 6 continued)

(v) [4 marks] Complete the following predicate to model the "Select Espresso" transition from the diagram on page 26.

```
pred SelectEspresso(s1, s2 : MachineState) {
  // s1 is before state, s2 is after state









}
```

(vi) [2 marks] Give an Alloy fact that ensures every MachineState is always part of exactly one ExecutionTrace.

****

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.

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.

Student ID: . . . .

(This page may be detached)

Appendix

Example Alloy code for the Monopoly board is given here for reference.

```
sig Name, Colour {}

abstract sig Building {}{
  some buildings.this
}
sig House extends Building {}
sig Hotel extends Building {}

fact {
  // There can be no more than 32 houses and 12 hotels at one time
  // For simplicity, we lower these values since they're too big for Alloy!
  #House <= 7
  #Hotel <= 2
}

sig Player {}

sig Property {
  name: Name,
  owner: lone Player,
  buildings: set Building,
  colourGroup : one ColourGroup,
}{
 // Up to four houses or one hotel in buildings
 #buildings <= 4
 some h : Hotel | h in buildings
 all h : Hotel | h in buildings implies #buildings = 1

 // built properties must have an owner
 some buildings implies one owner
}

sig Mortgaged in Property {}{ no buildings }

sig ColourGroup {
 colour: Colour,
}{
 // Every ColourGroup has a unique colour
 all cg : ColourGroup | cg.@colour = colour implies cg = this
```

Student ID: . . . .

```
 // ColourGroup Arity
 #~colourGroup >= 2 && #~colourGroup <= 3

 // You must own a whole ColourGroup in order to build
 all p : this.~colourGroup | (some p.buildings) implies
   (all p' : this.~colourGroup | p.owner = p'.owner)

 // buildings must be equally distributed
 all disj p, p' : this.~colourGroup, h: Hotel |
   (h in p.buildings) implies ((some h' : Hotel | h' in p'.buildings) ||
                                #p'.buildings = 4)

 all disj p, p' : this.~colourGroup |
   (no h : Hotel | h in p.buildings) implies (
   #p.buildings = #p'.buildings ||
   #p.buildings = #p'.buildings + 1 ||
   #p.buildings = #p'.buildings - 1 ||
   (#p.buildings = 4 && some h : Hotel | h in p'.buildings))
}

run {} for 4 but 2 ColourGroup, 8 Building, 6 int, exactly 1 Hotel
```