TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VUW VICTORIA UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2012

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.
Non-electronic foreign language dictionaries are allowed.
Calculators ARE NOT ALLOWED.
No other reference material is allowed.

Question	Topic	Marks
1.	Use Cases and Requirements	30
2.	Objects and Classes	30
3.	Writing Invariants	30
4.	Alloy I	30
5.	State Machines	30
6.	Alloy II	30
	Total	180

Student ID: .....

Question 1. Use Cases and Requirements [30 marks]

(a) [3 marks] Perform a *textual analysis* on the following description, by carefully and neatly underlining key verb phrases that could lead to candidate use cases.

The OldeWorlde supermarket chain is preparing a web site to support on-line shopping. After registering a user account, shoppers can access the website to choose products to buy. Shoppers can list products from a given product category (fruit, vegetables, chocolate, fizzy drink) or they can type in a name or a bar code. Once they've finished shopping, they must use the web site to "check out". They can pay with a credit card or bank account number which must be set up when their account is registered.

Products are collected from supermarket shelves by supermarket employees called "buying assistants". Using an iPad, buying assistants can request the next shopping list, get a shopping trolley, and then go through the store filling up the trolley, and putting the shopping directly into bags. Buying assistants don't go to the checkout because everything is listed on their iPad.

The shopping is delivered to each shopper's registered address by delivery trucks. A truck picks up a bunch of orders that are nearby to each other, delivers the shopping.

It's really important that the supermarket manager can be sure what's going on. Managers need to know how many orders are waiting to be filled by buying assistancts, and how many orders are waiting to be delivered by the trucks.

Student ID: .....

(b) [3 marks] Write a *use case description* for the **Register Account** use case.

(c) [3 marks] Give characteristics for the **Buying Assistant** actor.

(d) [6 marks] Name **three** systemic requirements for this system — for each requirement, write a brief sentence justifying its importance.

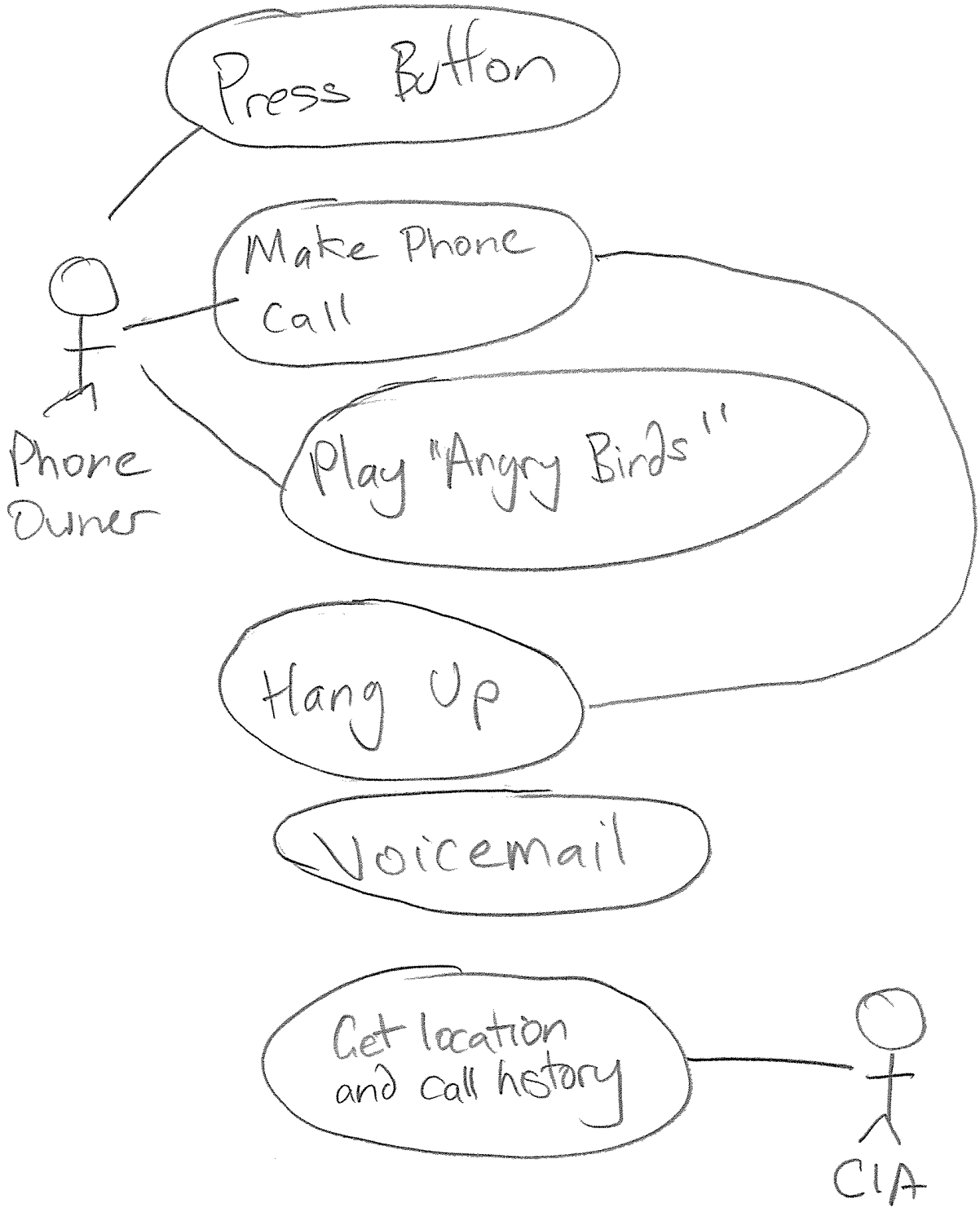
1.

2.

3.

Student ID: .....

(e) [5 marks] The following use-case diagram for a new mobile phone was drawn by a student intern before they left to study for a Masters' degree in Software Engineering.



Student ID: . . . . . .

Circle **five distinct** problems in this diagram. For each problem, number it and briefly describe why it is a problem.

1.

2.

3.

4.

5.

(f) [10 marks] Handling Errors

A budget airline has a website for booking tickets. Users can use the **Show Flight** usecase at any time, but before they can book and pay for flights with the **Book Flight** usecase, they must perform the **Log In** usecase.

(i) [2 marks] Briefly sketch a use-case diagram that uses *inclusions* in order to model users logging in to the website.

(ii) [2 marks] Briefly sketch a use-case diagram that uses *preconditions* in order to model users logging in to the website.

(iii) [2 marks] Briefly sketch a use-case diagram that uses *extensions* in order to model users logging in to the website.

(iv) [4 marks] Explain which solution you prefer, and why.

Student ID: .....

Question 2. Objects and Classes [30 marks]

(a) [15 marks] Based on the description from Question 1 on Page 2, draw a *well-designed* **class diagram** to model the OldeWorlde online shopping system. This should contain at most 6 classes and 10 attributes and use inheritance and associations where appropriate.

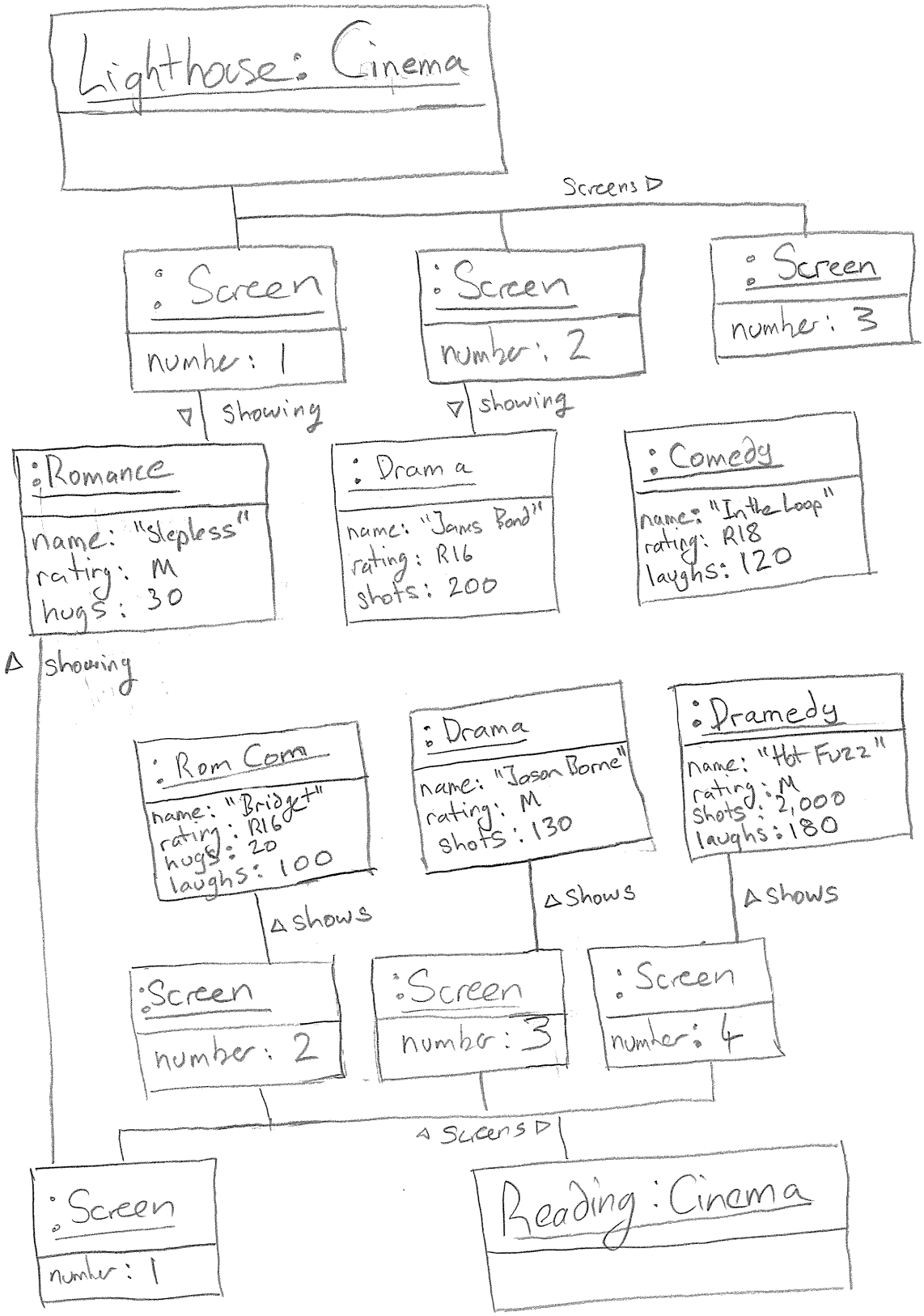
Student ID: .....

(b) [5 marks] In the box below, draw an *object diagram* consistent with your class diagram which captures the following scenario:

> "Jian, a SWEN102 student, ordered 3 cans of Red Bull and 2 packets of Salt-and-Vinegar chips. This order is being delivered to his delivery address: 'Cotton Building room CO145'. Meanwhile, Terri has just ordered one copy of the 'Guardian Weekly', but she hasn't finished shopping yet."

Student ID: .....

(c) [10 marks] Consider the object diagram below and draw a corresponding *well-designed* **class diagram** on the facing page.



(Question 2 continued on next page)

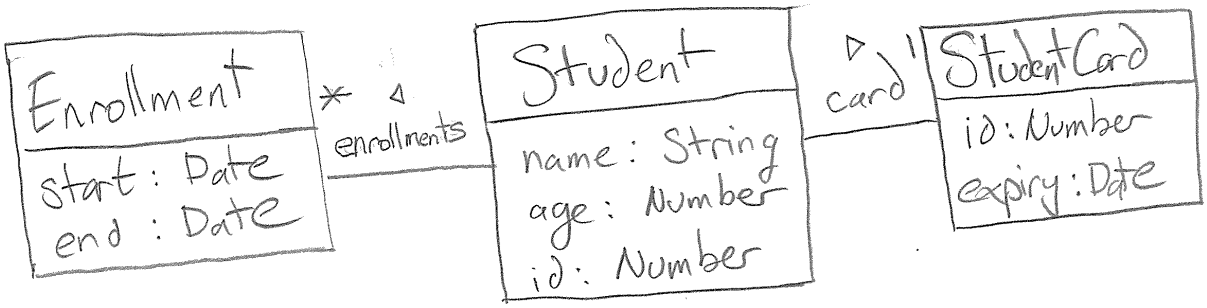
Student ID: . . . . . .

(Question 2 continued)

Student ID: .....

Question 3. Writing Invariants [30 marks]

(a) [10 marks] Given the following class diagram, translate the following invariants into formal notation.



(i) [1 mark] Students must be over 18.

(ii) [1 mark] A student has exactly one student card.

(iii) [2 marks] A student's card id number is their student number.

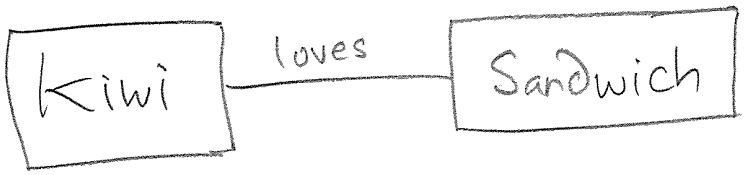
(iv) [3 marks] A student card's expiry date is their last enrollment end date.

(v) [3 marks] Student's enrollments cannot overlap — for any given date there can be only one enrollment.

Student ID: .....

(b) [10 marks]

We are often told that *"Every Kiwi loves a Marmite Sandwich"* — but what does this statement really mean? Given the following model:



(a) Write **five distinct alternative** formal invariants (alloy facts) that could reasonably translate this statement. (b) Translate each alternative back into more precise English.

1. (a)

   (b)

2. (a)

   (b)

3. (a)

   (b)

4. (a)

   (b)

5. (a)

   (b)

(c) [10 marks]

Imagine you have joined a software engineering company for your summer work placement. Write down the arguments you will use to convince your boss that your team should write invariants about their designs. As a software engineer, you should explain the strenghts and benefits of writing invariants, but also the weaknesses and costs.

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 4. Alloy I [30 marks]

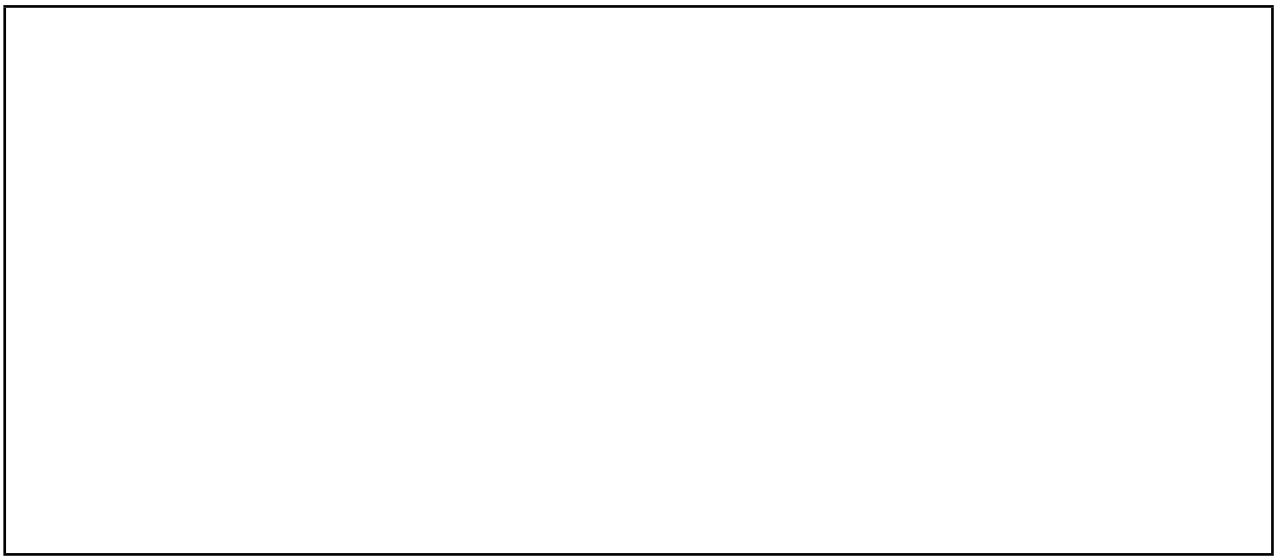
Consider the following description of an aeroplane managing model:

```
sig EngineTester{
  testedEngine:lone Engine
  }

abstract sig Aeroplane{
  engines: set Engine
  }

sig BigAeroplane extends Aeroplane{
  }{
  some disj e1,e2,e3,e4:Engine| e1+e2+e3+e4=engines
  }

sig LittleAeroplane extends Aeroplane{
  }{
  some disj e1,e2:Engine| e1+e2=engines
  }

sig Engine{lastRevision:Date}{
  lone this.~engines + this.~testedEngine
  }

sig Date{}

sig CanFly in Aeroplane{}

run A{lone EngineTester}
run B{some BigAeroplane and some LittleAeroplane} for 10
```

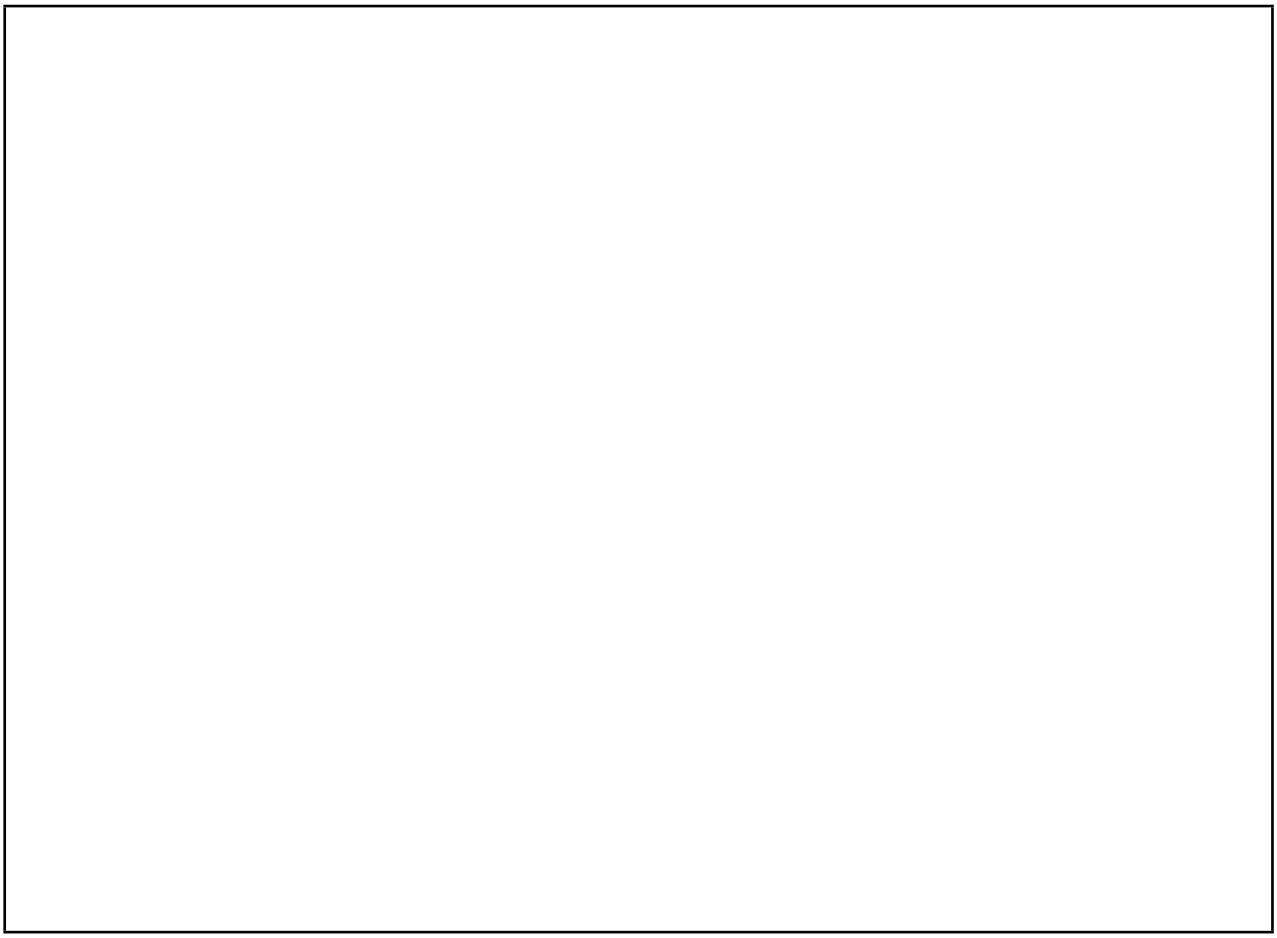
(a) [3 marks] Write in English the meaning of the constraint on **sig** LittleAeroplane.

Student ID: .....

(b) [3 marks] Draw a diagram corresponding to a possible instance found by the command "**run** A{**lone** EngineTester}"



(c) [5 marks] Draw a diagram corresponding to a possible instance found by the command "**run** B{**some** BigAeroplane **and some** LittleAeroplane} for 10".



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: .....

(d) [14 marks] Draw a UML class diagram representing the Alloy model

(e) [5 marks] Write in English the meaning of the constraint on **sig** Engine.

Student ID: .....

Question 5. State Machines [30 marks]

Consider the following description the simple *automatic cleaner robot Toribash*:

> "Toribash will roam around on the floor *collecting dust*, until the battery power goes down to *low*. When this happens Toribash will *return to the base* to automatically *recharge* its battery. As soon as the battery is *full*, the Toribash will start to roam around again. In the unlikely event Toribash does not manage to reach the *base* in time and the battery become *exhausted*, Toribash will *deactivate* itself.

```
enum RobotState{Recharging,CollectingDust,ReturnToBase,Deactivated}
one sig ExecutionTrace{states:seq RobotState}{
  states[0]=Recharging
  all s1,s2:states.inds| s2=s1+1 implies
    fullyCharged[s1.states,s2.states] or
    cleaningAround[s1.states,s2.states] or
    fullToLow[s1.states,s2.states] or
    reachBase[s1.states,s2.states] or
    endBattery[s1.states,s2.states] }
pred fullyCharged[r,r':RobotState]{ r=Recharging and r'=CollectingDust }
pred cleaningAround[r,r':RobotState]{ r=CollectingDust and r'=CollectingDust }
pred fullToLow[r,r':RobotState]{ r=CollectingDust and r'=ReturnToBase }
pred reachBase[r,r':RobotState]{ ... }
pred endBattery[r,r':RobotState]{ r=ReturnToBase and r'=Deactivated }
```

(i) [7 marks] Provide a graphical representation of the alloy model as a state machine.

(Question 5 continued on next page)

(Question 5 continued)

(ii) [2 marks] Complete the code of the predicate reachBase.

(iii) [5 marks] Write a **run** command showing an execution trace where the robot reaches the *Deactivated* state. Draw a diagram corresponding to a possible instance found by this run command.

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.

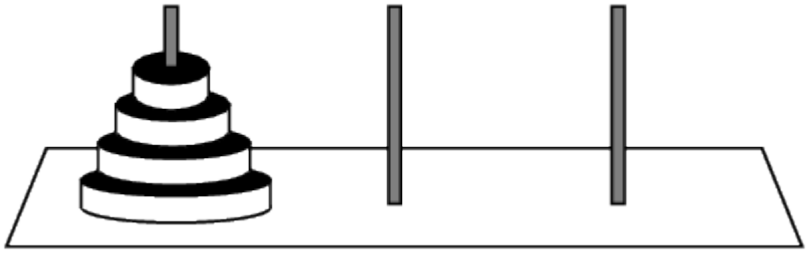
(iv) [8 marks] Write a **run** command producing an execution trace where the robot manages to recharge itself more than once. Draw a diagram corresponding to a possible instance found by the **run** command.

(v) [8 marks]

How would you use Alloy to find out if the robot can leave the Deactivated state? Give any Alloy commands you would use. If those commands produce diagrams, draw a possible instance; if not, say why not.

Student ID: .....

Question 6. Alloy II [30 marks]



Consider the following alloy description of a valid Towers of Hanoi game state:

```
sig Disk{size:Int}{size>0}
abstract sig Pole{disks:seq Disk}
one sig Pole1,Pole2,Pole3 extends Pole{}

fact P1{ all p:Pole, i,j:p.disks.inds| i>j implies p.disks[i].size>p.disks[j].size }
fact P2{ all disj p1,p2:Pole|  no p1.disks.elems & p2.disks.elems }
fact P3{ all p:Pole | not p.disks.hasDups }
fact P4{ all p:Pole, i:p.disks.elems.size| i>=0 }
fact P5{ some p:Pole| no p.disks.elems implies Disk=(Pole-p).disks.elems}
fact D1{ all i:Int| lone i.~size}
fact D2{ Disk = Pole.disks.elems }
```

(a) [6 marks] Consider the following *textual description* of the Towers of Hanoi game:

> There are three poles, each pole contains some disks of various (positive) sizes. All the disks are piled up the poles, using the pole as a support. Every disk lies on the top of a bigger disk.

There is a difference between what is required in the *textual description* and what is *modelled in the alloy code*. Indeed no object diagram can satisfy the model in the *textual description*. Minimally modify the *textual description* to describe the *model in the alloy code*.

Student ID: .....

(b) [4 marks] Write a **run** command showing the three Poles with at least 3 Disks each.

(c) [6 marks] Provide better names for the Alloy facts P1, P2, D1, D2.

P1

P2

D1

D2

(d) [8 marks] Some **fact**s are redundant, and can be converted into **check**s, without changing the set of allowed object diagrams.

Write in the following box the names of the redundant facts and explain why they are redundant.

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: .....

(e) [6 marks] Write a predicate **pred** isOkToMove[p1,p2:Pole]{...} that checks if moving the top disk from p1 to p2 would produce another valid Hanoi Tower model.

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.

****