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

EXAMINATIONS — 2007

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.	Functional and Systemic Requirements	30
3.	Object and Class Diagrams	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.

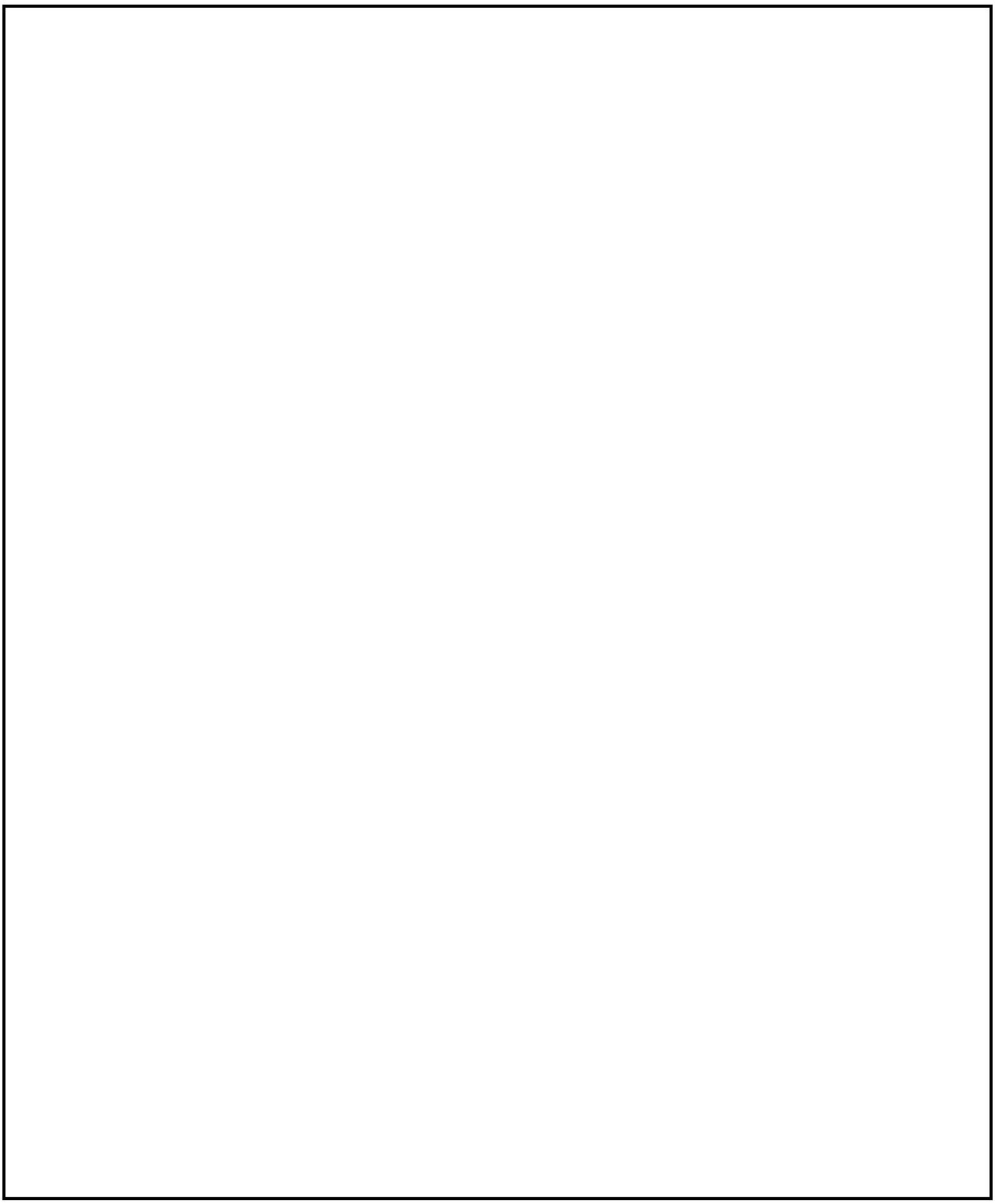
> The border control system checks the passport of every traveller arriving at New Zealand's airports.
>
> The system is mainly used by customs officers. After logging in, an officer can register an arriving traveller by entering their nationality and passport number into the system. Officers may use a scanner to read the passport, or may enter the details by typing on a keyboard. Then, senior customs officers can get reports on who has arrived on a particular day.
>
> Frequent travellers with New Zealand passports may visit a special office in the airport, where digital photographs are taken of their fingerprints. The frequent traveller can then enter the country through an automatic gate that scans their fingerprints.
>
> Police officers with a warrant from a judge may request that a passport is blocked by the system, to stop the passport holder leaving the country.
>
> Finally, system administrators must add and delete information about the customs officers, police officers, and anyone else who is permitted to use the system.

Student ID: .....

(b) [6 marks] The description text is incomplete. Give **two** questions you would ask users or clients to clarify these requirements. If necessary, also give the answer you have **assumed** to this question.

Student ID: .....

(c) [12 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.



(d) [9 marks]

Characterise two actors in the system in terms of the level of domain knowledge, system knowledge and frequency of interaction, writing one short sentence for each part.

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

Write a short persona for a third actor in the system (not one of the actors above)

Student ID: .....

Question 2. Functional and Systemic Requirements [30 marks]

The following text describes a computerised parking system:

> The Computronic Park-O-Matic is an electronic parking system designed to prevent citizens from abusing their city's car parks.
>
> A parking terminal is placed near every car park. If citizens can find somewhere to park their cars, they can pay cash by entering the carpark number into a machine, paying ten dollars an hour for up to three hours parking, then printing a receipt that must be left in their car. Alternatively, citizens can pay by SMS text messages — texting "PAY" and the carpark number to the special number 7275 (PARK) — the machine then issues a receipt.
>
> Citizens who park often can purchase a season ticket by giving their credit card number to a website. They are then sent a season ticket by regular mail. Once they have received their season ticket, they can park by entering the season ticket number into the parking machine. Season tickets holders can display their parking history via a web interface — a list of where and when they have parked and how much credit they have left.
>
> Finally, council parking wardens can use a mobile phone to get a list of all parking spaces in their area that are not currently paid for.

Student ID: . . . . . .

(a) [20 marks] Draw **essential use case cards** for the following **five** use cases in this system.

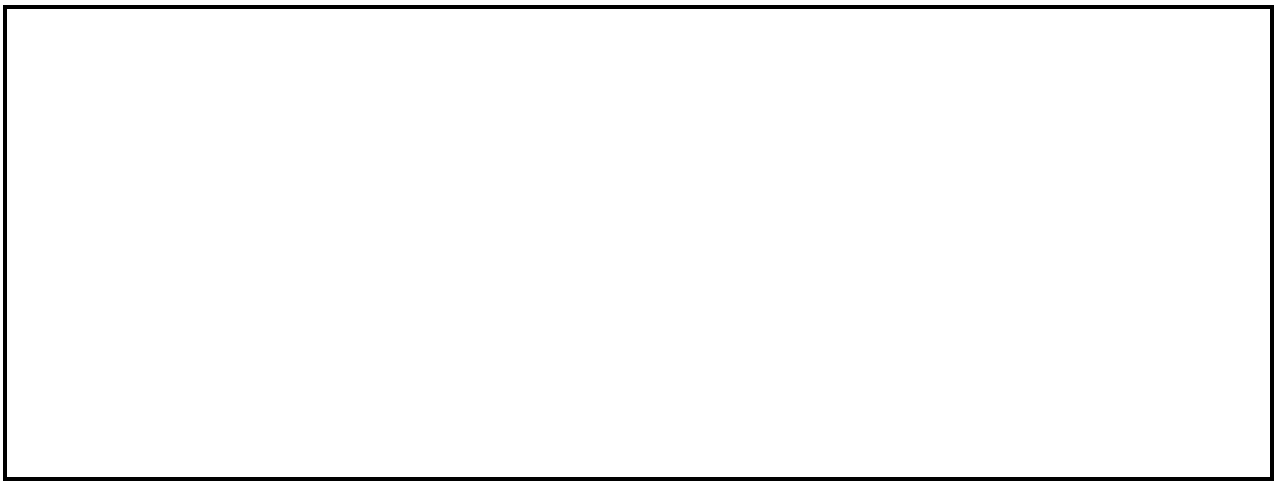
Pay Cash for Park

Buy Season Ticket

Display Parking History

Student ID: . . . . . .

Pay with Mobile Phone for Park



List Unpaid Parking Spaces in Area

Student ID: . . . . . .

(b) [10 marks]

Briefly describe **5** important systemic requirements that could apply to the parking system.

1.

2.

3.

4.

5.

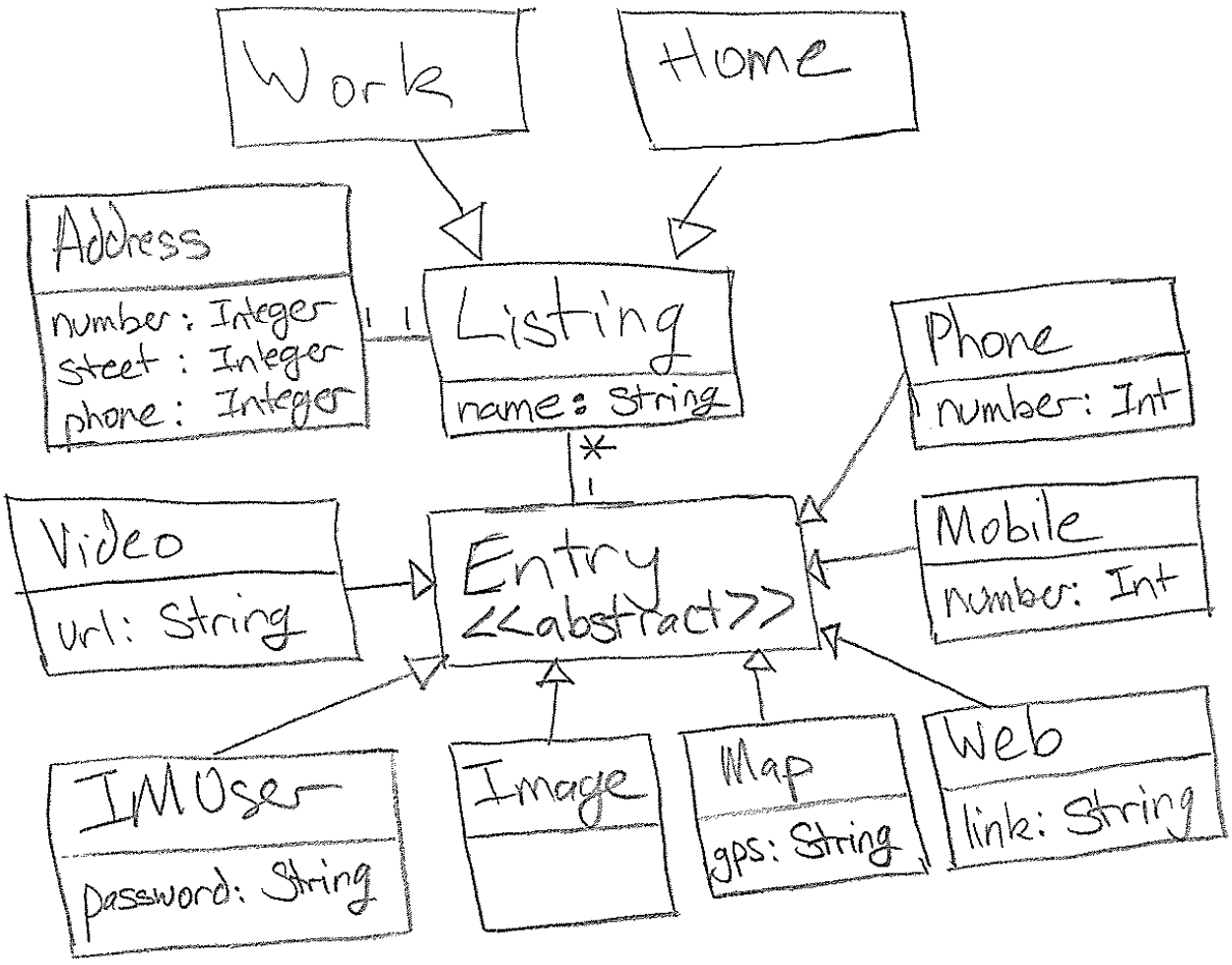
Student ID: .....

Question 3. Object and Class Diagrams [30 marks]

(a) [15 marks]

The class diagram below is supposed to model a telephone directory system:

> The Telecon SuperDirectory project hopes to store all kinds of directory information in a single place. Directory listings must have a name and an address, but may then have one or more phone numbers, mobile numbers, email addresses, web addresses, instant messaging (IM) usernames, images, or map locations. Work listings may also have a mailing address, while home listings just have one address.



Student ID: . . . . . .

Circle and number **seven distinct** problems in the class diagram.
Describe briefly why each problem is a problem.

1.

2.

3.

4.

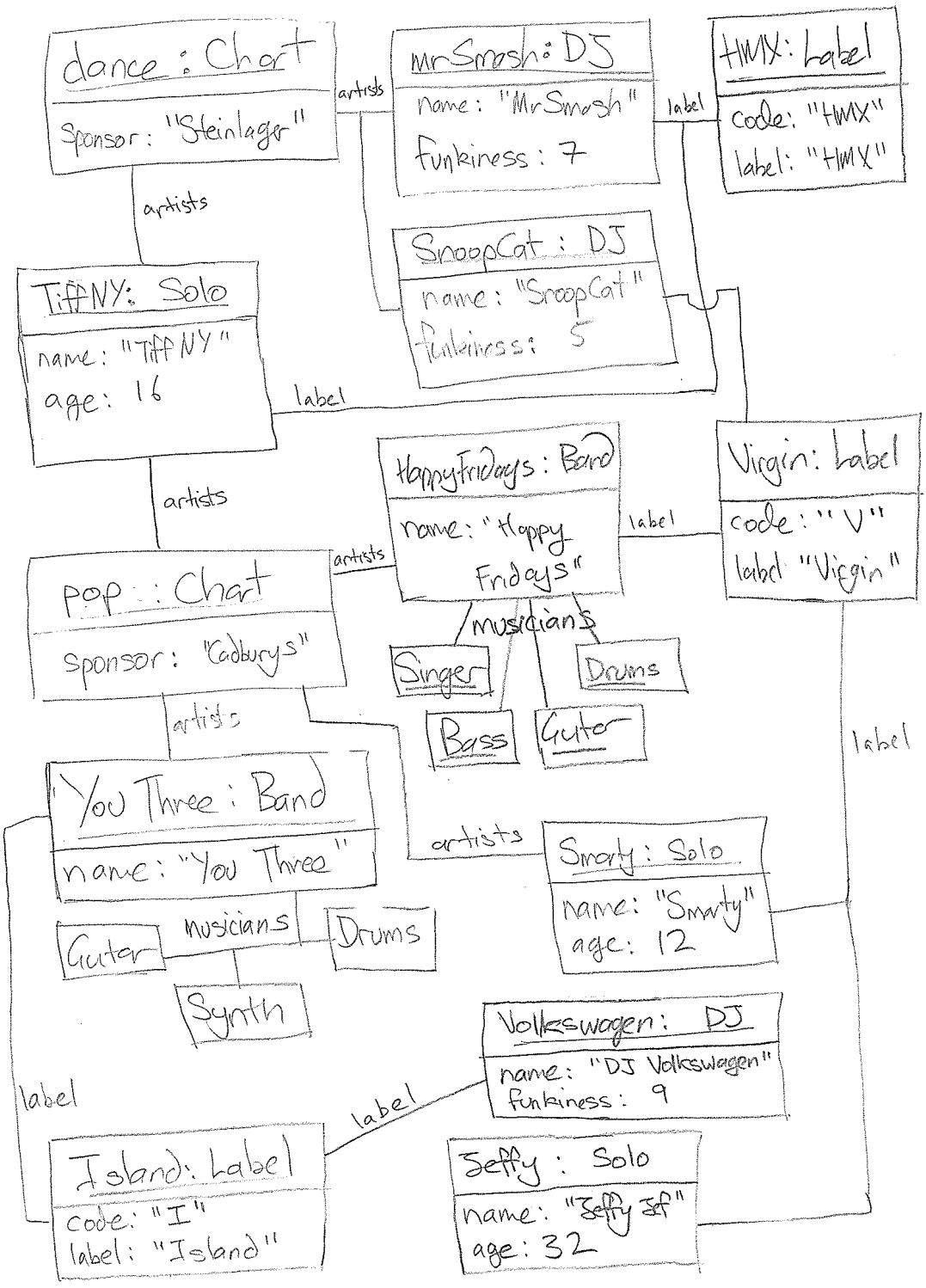
5.

6.

7.

Student ID: .....

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

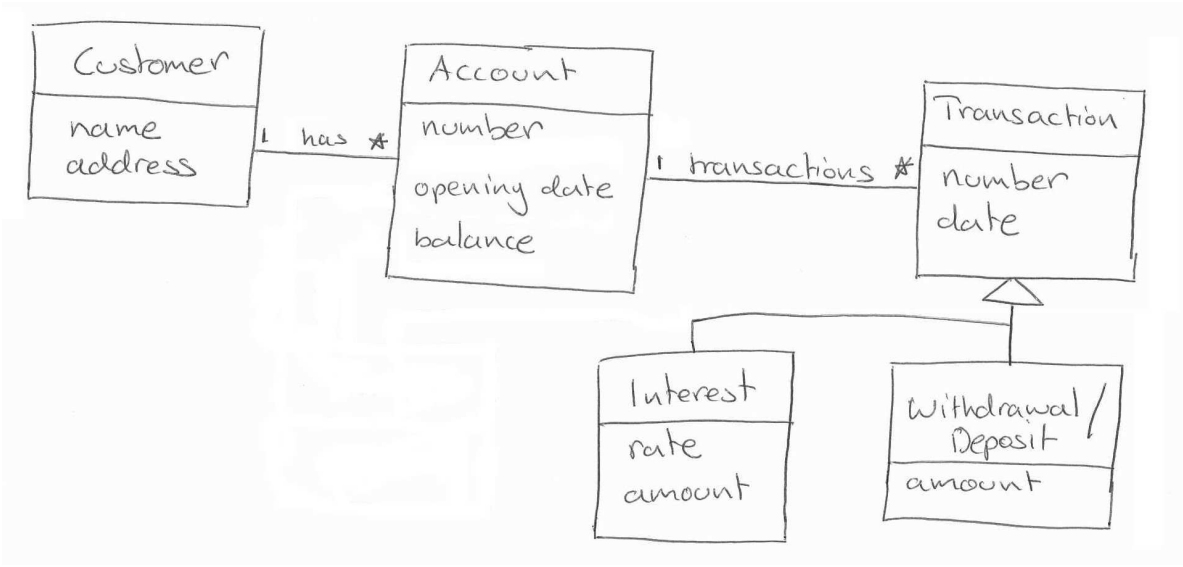


Student ID: .....

Question 4. Writing Invariants [30 marks]

Consider the following description for a *bank account system*, which is made up of some *text* and a *class diagram*:

> "The bank account system stores details of customers, accounts and transactions. Every account has an *account number*, such that no two accounts have the same number. The date an account was opened and the current balance are recorded.
>
> A transaction is either a *withdrawal/deposit* or a credit to the account for *interest* accumulated on the balance. The date when each transaction took place, and a unique transaction number are recorded. Furthermore, the amount of interest credited is never negative, and the interest rate is stored as a percentage."



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

No two accounts have the same number.

No two transactions have the same transaction number.

(Question 4(a) continued)

Interest credited is never negative

Interest rate ≥ 0 and ≤ 100

Every transaction on an account must take place after the account was opened

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

```
all a1,a2:Account | a1.number = a2.number implies a1=a2
```

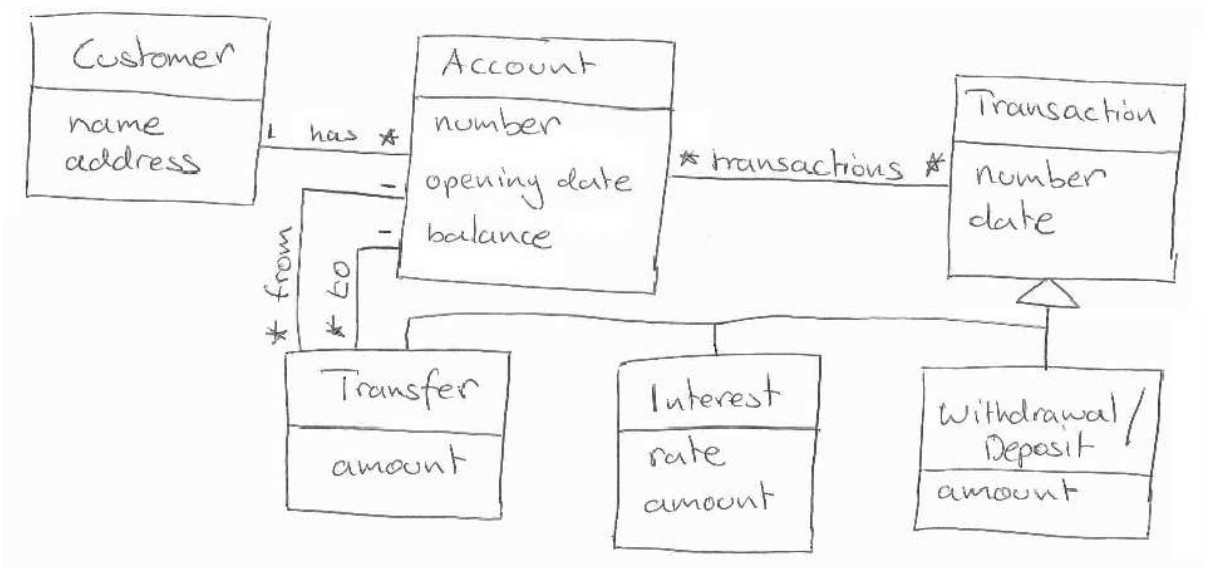
```
no disj t1,t2:Transaction | t1.number = t2.number
```

```
all i:Interest | i.amount >= 0
```

```
all i:Interest | i.rate >= 0 && i.rate <= 100
```

Student ID: .....

The account system was extended to support *transfers* from one account to another, as shown in the following class diagram and global invariants:




```
all t:Transfer | t.to != t.from

all a:Account, t:Transfer | t in a.transactions implies (a=t.to || a=t.from)

all a1,a2:Account,t1:Transfer | (t1 in a1.transactions &&
                                 (a2 in t1.from+t1.to)) implies
                                  t1 in a2.transactions
```

(c) [10 marks] Translate the three invariants in Alloy given above into written English:

No transfer can be from one account to the same account!

For every transfer on an account, the transfer must be either to or from that account

Every transfer must be recorded as a transaction on both accounts involved

Student ID: .....

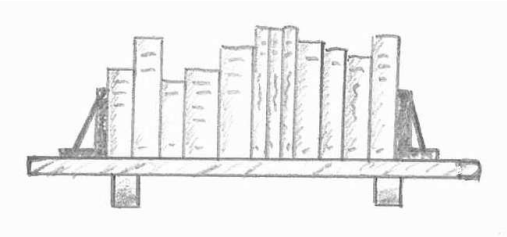
SPARE PAGE FOR EXTRA ANSWERS

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

Student ID: . . . . . .

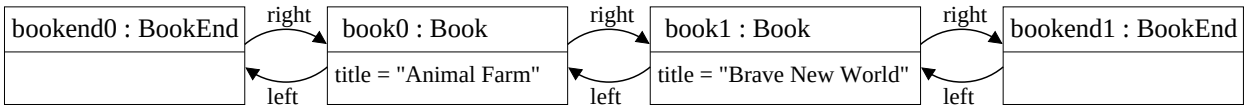
Question 5. Using Alloy [30 marks]

Consider the following description of a bookshelf:



> "Books are arranged on a shelf, such that every book has one book or bookend to its left and one to its right. Bookends always have a book or a bookend on one side and nothing on the other side. Books have titles, but bookends don't!"

(a) [5 marks] In the box below, draw an object diagram that is *consistent* with the description of a bookshelf. The diagram should include at least two `Book`s and one `Bookend`.



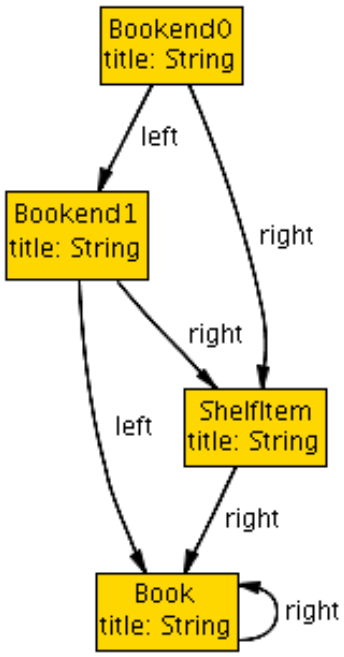
Student ID: .....

An *incorrect* implementation of a book shelf in Alloy and an *object diagram* generated using the command "`run {} for 4`" are given below:

```
sig String {}

sig ShelfItem {
 left : lone ShelfItem,
 right : lone ShelfItem,
 title : String
}

sig Book  extends ShelfItem { }
sig Bookend  extends ShelfItem { }
```



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

1) Bookend0 has something on its left and its right!
2) Bookend0 is left of Bookend1, but Bookend1 is right of ShelfItem!
3) ShelfItem should not be allowed, since it's an abstract concept
4) Book is right of itself!
5) Bookend0 has a title!

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: .....

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

Add following class invariant to `Bookend`: `left=None || right=None`

Note, this could also be written as: `#(left+right)=1`

Add following class invariant to `ShelfItem`: `left = ~ right`

Make `ShelfItem` abstract like so: `abstract sig ShelfItem ...`

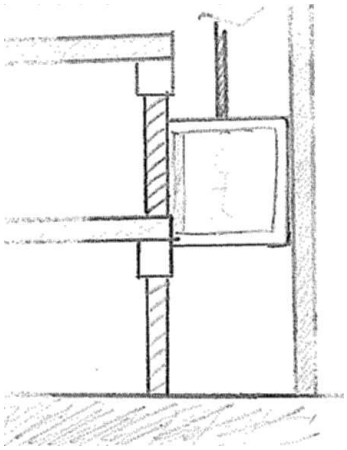
Add following class invariant to `ShelfItem`: `this not in this.^ left`

Remove `title` from `ShelfItem`, and place it in `Book` instead

Student ID: .....

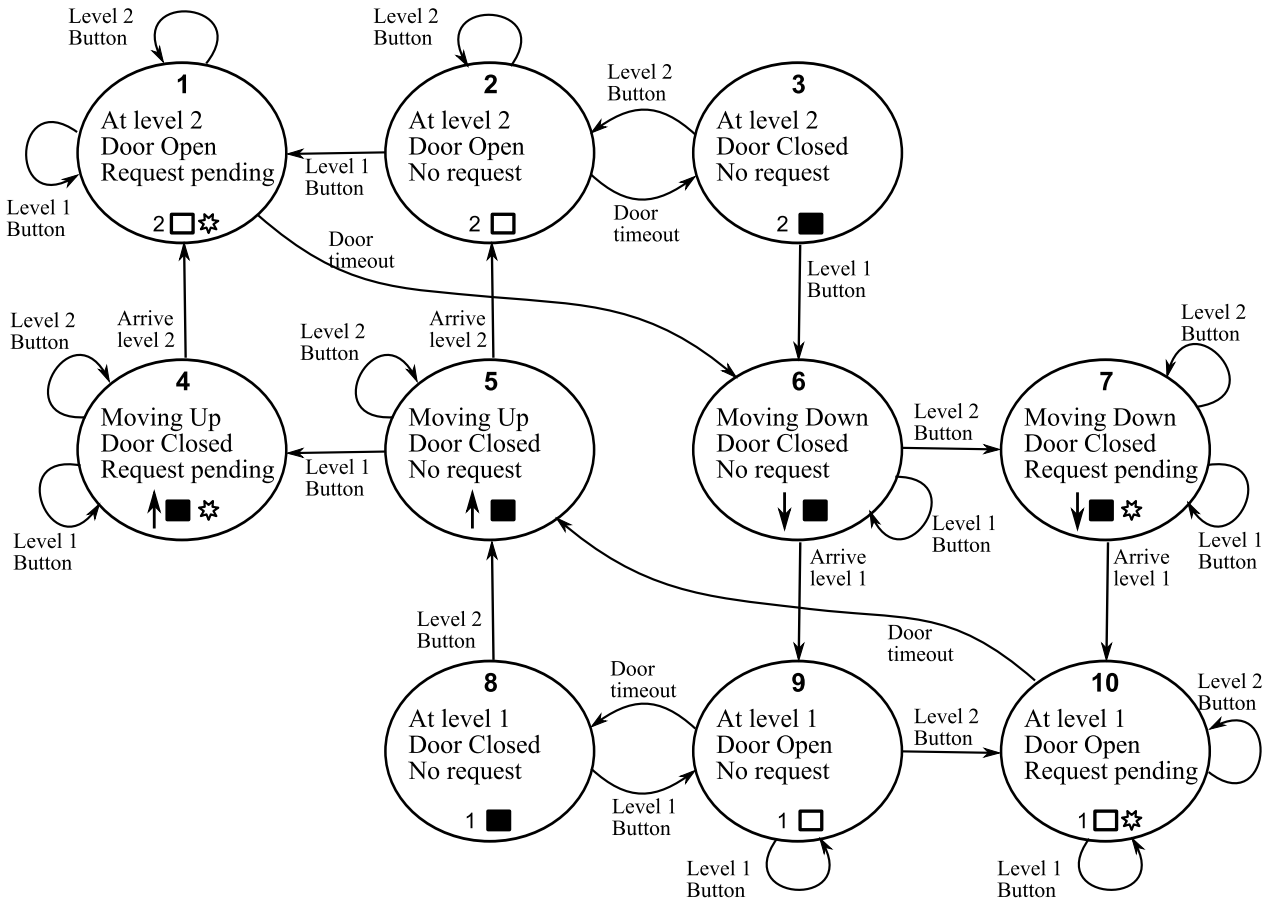
Question 6. State Machines [30 marks]

Consider the following description of a *hotel lift system*:



"The hotel has two floors, designated *level 1* and *level 2*. The lift transports *guests* between floors. There is a *lift request button* situated on each level; pressing this requests the lift to come to that floor. The lift has a sensor that detects when it has arrived on a particular floor. When it arrives, the lift doors open for a fixed time, after which they automatically close."

A *state machine diagram* for the hotel lift system has been provided:



Student ID: .....

(a) [5 marks] A *state machine diagram* consists of *states* and *transitions*. In your own words, describe what a *state* is, using the lift system as an example.

A state describes the current arrangement of things in the system at a particular moment in time. For example, in the lift system, a state describes the current state of the lift doors, the lift motor and the request switch.

(b) [10 marks] Each state in the state machine diagram given for the hotel system has a unique number. Use these numbers when answering this question.

(i) In which state(s) is the lift doing nothing, other than to wait for the next request?

3,8

(ii) In which state(s) might people be getting on or off the lift?

1,2,9,10

(iii) In which state(s) is the lift moving between floors?

4,5,6,7

(iv) In which state(s) might someone be waiting for the lift to finish responding to an earlier request?

1,4,7,10

(v) In which state(s) is the lift about to begin descending to Level 1?

1

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.

(c) [8 marks] Provide a suitable *execution trace* for each of the following scenarios. Your execution trace may start from whichever state you chose.

"Jane requested the lift on Level 1 and the doors opened immediately."

Level 1 Closed No Req	$\xrightarrow{\textit{Level 1 Button}}$	Level 1 Open No Req

"John Requested the lift on Level 1. He could hear the lift coming down from Level 2 and it soon arrived."

Level 2 Closed No Req	$\xrightarrow{\textit{Level 1 Button}}$	Down Closed No Req	$\xrightarrow{\textit{Arrive Level 1}}$	Level 1 Open No Req

"Sandy exited the lift on Level 2. When the doors closed, she heard the lift going down to Level 1."

Level 2 Open Req Pending	$\xrightarrow{\textit{Timeout}}$	Down Closed No Req

"James requested the lift on Level 1. He could hear the lift was going up to Level 2 and knew he would have to wait."

Up Closed No Req	$\xrightarrow{\textit{Level 1 Button}}$	Up Closed Req Pending

Student ID: .....

(d) [7 marks] The design given for the hotel lift system has a *serious flaw*.

(i) What important use case is not captured in the design?

There are no buttons inside the lift. So, somebody getting into the lift cannot request that it go up/down a level!

(ii) Suggest a simple way in which the design could be modified to accommodate the use case identified in **(i)**.

We add request buttons inside the lift itself. These would work in much the same way as the request buttons on either level

****

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 Player {} {}
abstract sig Boolean {}
lone sig True, False extends Boolean {}

abstract sig Building {}
sig House extends Building {}
sig Hotel extends Building {}

sig Property {
  name: String, owner: lone Player, buildings: set Building,
  mortgage : Boolean, colourGroup : one ColourGroup
}{
 buildings in Property some -> set Building
 #buildings <= 4
 some h : Hotel | h in buildings
 all h : Hotel | h in buildings implies no h' : House | h' in buildings
 mortgage = True => no buildings
 some buildings implies one owner
}

sig ColourGroup { colour: String
}{
 all cg : ColourGroup | cg.colour = colour implies cg = this

 #~colourGroup >= 2 && #~colourGroup <= 3

 all p : this.~colourGroup | (some p.buildings) implies
   (all p' : this.~colourGroup | p.owner = p'.owner)

 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))
}
```