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

EXAMINATIONS — 2011

MID-TRIMESTER TEST

SWEN 102
Introduction to Software Modelling

Time Allowed: 50 minutes

Instructions: There are 50 possible marks on the exam.
Answer all questions in the boxes provided.
Every box requires an answer.
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.	Alloy I	25
2.	Alloy II	25
	Total	50

Student ID: .....

Question 1. Alloy I [25 marks]

Consider the following description of *blocks*:

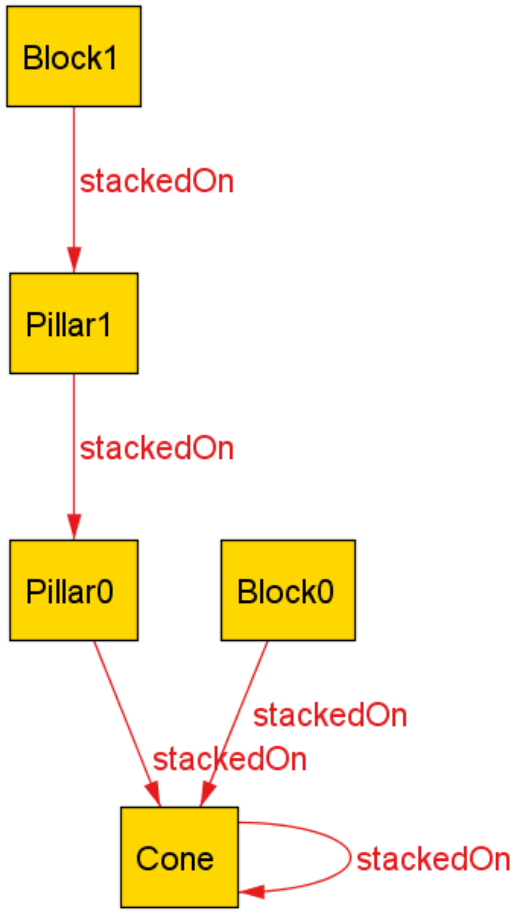
> "Blocks are 3-dimensional objects which can be *stacked* on top of each other. There are two kinds of block: *pillars* and *cones*. A block can be stacked on at most one block, and can be under at most one block. No block can be stacked upon a cone and no block can be stacked upon itself."

An *incorrect* Alloy model of the blocks system and an *object diagram* of that model are given below:

```
sig Block {
  stackedOn : lone Block
}

sig Pillar extends Block {}
sig Cone extends Block {}

run {} for 7 but exactly 5 Block, 1 Cone
```



The above object diagram was generated using the given `run` command.

(Question 1 continued on next page)

Student ID: .....

(Question 1 continued)

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

`Pillar$1.stackedOn` ={`Pillar$0`}
`Pillar$0.~stackedOn` ={`Pillar$1`}
`Pillar$1.stackedOn + Pillar$0.stackedOn` ={`Pillar$0,Cone`}
`Pillar$1.*stackedOn` ={`Pillar$1,Pillar$0,Cone`}

(b) [8 marks] Identify four distinct ways in which the Alloy model given is *inconsistent* with the description given for blocks.

1)A block should be either a Pillar or Cone. However, the model includes blocks (such as Block$1) which are neither.
2)A block cannot be stacked upon a cone. However, the model includes blocks (such as Pillar$0) which are stacked on a cone.
3)A block cannot be under more than one block. However, the model includes a block (i.e. Cone) which is under two other blocks (i.e. Pillar$0 and Block$0).
4)A block cannot be stacked upon itself. However, the model includes a block which is stacked upon itself (i.e. Cone).

(Question 1 continued on next page)

Student ID: .....

(Question 1 continued)

(c) [12 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)The declaration of `Block` should be made abstract, as follows:

```
abstract sig Block { ...  }
```

This would prevent a block from existing which is neither a Pillar nor Cone.

2)The declaration of `Cone` should be modified to prevent blocks from being stacked upon it, as follows:

```
sig Cone extends Block {}{ no this.~stackedOn }
```

This ensures that no blocks can be stacked on any `Cone`.

3)The declaration of `Block` should be modified to restrict the number of blocks that can be stacked upon it, as follows:

```
abstract sig Block { stackedOn: lone Block}{ lone this.~stackedOn }
```

This ensures that at most zero or one blocks can be stacked on any `Block`.

4) The declaration of `Block` should be modified to prevent itself from being in the set of blocks reachable via the `stackedOn` relation, as follows:

```
abstract sig Block { stackedOn: lone Block}{
    lone this.~stackedOn
    this not in this.^stackedOn
}
```

This ensures that no block can be in its own transitive closure.

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 2. Alloy II [25 marks]

Consider the following simple model for a *library system*:

```
sig string {}
enum Bool { No, Yes }

sig Borrower {
 books : some Book,
 limit : Int
}
sig Book {
 title : string,
 closedReserve : Bool
}
```

(a) [10 marks] Translate each of the following statements into an Alloy fact.

1) A borrower's limit is a positive integer.

```
fact { all b :  Borrower | b.limit >= 0 }
```

2) A borrower cannot borrow more books than his/her limit.

```
fact { all b :  Borrower | #b.books <= limit }
```

3) No two books can have the same title.

```
fact { all disj b1,b2 :  Book | b1.title != b2.title }
```

4) No book can be borrowed by more than one borrower.

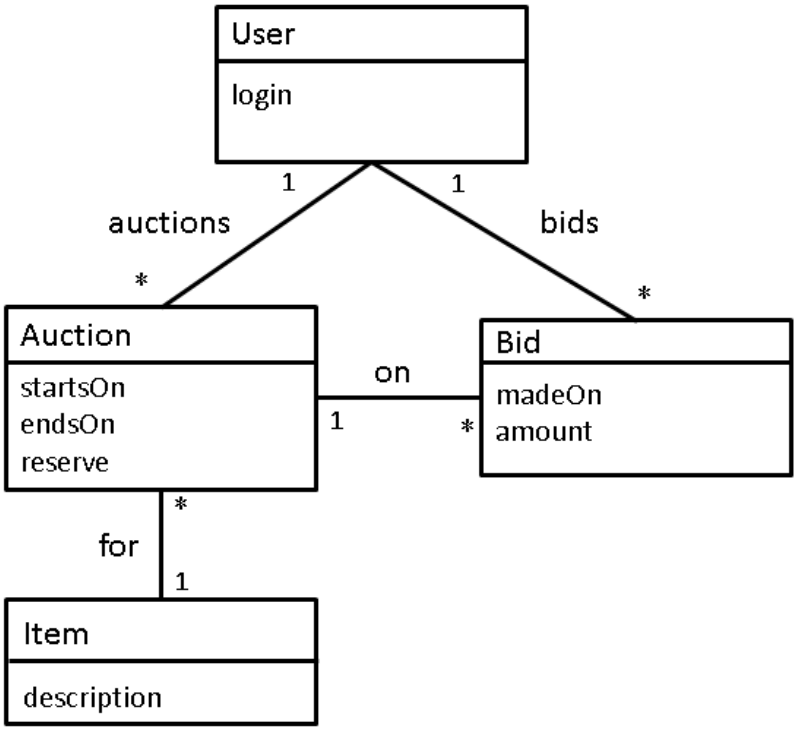
```
fact { all b :  Book | lone #b.~books }
```

5) No book on closed reserve can be borrowed.

```
fact { all b :  Book | b.closedReserve = Yes implies no #b.~books }
```

Student ID: .....

Consider the following class diagram for an *auction system*, and the additional rules given below.



- The amount bid on an auction cannot be negative.
- The reserve for an auction cannot be negative.
- An auction cannot end before it has begun.
- A bid must occur between an auction's start and end time.
- A user cannot bid on one of their own auctions.
- A item cannot be in more than one auction at the same time.

Student ID: .....

(b) [15 marks] In the box below, provide an Alloy model for the auction system. You should use Ints to represent times, and you may assume unlimited bit-width.

```
sig string {}

sig User {
  login:  string,
  auctions:  set Auction,
  bids :  set Bid
}

sig Auction {
  reserve :  Int,
  startsOn :  Int,
  endsOn :  Int,
  item:  one Item
}{
  one ~auctions
  // auction reserve cannot be negative
  reserve >= 0
  // auction cannot end before it has begun
  startsOn < endsOn
}

sig Bid {
  amount :  Int,
  madeOn :  Int,
  on :  one Auction
}{
  one  bids
  // bid amount cannot be negative
  amount >= 0
  // bid placed between auction start and end time
  madeOn >= on.startsOn
  madeOn <= on.endsOn
  // a user cannot bid on their own auction
  this. bids != on. auctions
}

sig Description {}

sig Item { description :  Description}{
  all disj a1,a2:this.~item | a1.endsOn<a2.startsOn or a2.endsOn<a1.endsOn
}
```

****

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
```