

CSE4213 Examination
June 2006
Instructions to Candidates

1. Examination time 2 hours
2. Answer all questions
3. There are 10 questions
4. Each question is worth 12 marks
5. Total marks 120
6. Calculators are not allowed
7. Use left hand page for rough working; this page will NOT be marked unless explicitly requested.
8. The *Concise Summary of the B mathematical toolkit* is supplied.

1. Give 3 attributes concerning non-determinism and its role in the B Method. For each attribute, give a corresponding example that demonstrates the attribute.

(a)

(4 marks)

(b)

(4 marks)

(c)

(4 marks)

(d) If your name is Julian, give a complete B implementation of Gregorian Simple Dates.
(For Anonymous Feedback readers only.)

(0 marks)

2. The B Toolkit uses abstract machines in specifying software behaviour. Explain how, paying particular attention to the notion of *state* and *operations* over the state.

(12 marks)

3. (a) Explain the role of preconditions in B specifications.

(3 marks)

(b) Explain the role of proof obligations in a B specification.

(3 marks)

(c) Abstract machine specifications in B do not allow the use of the sequential substitution operator “;” (semicolon). Explain why.

(3 marks)

(d) In the substitution **CHOICE G OR H END**, the Generalised Substitution Language form is $G[]H$, and the substitution $[G[]H]R$ can be rewritten as $[G]R \wedge [H]R$. Explain why.

(3 marks)

4. From the **SimpleBank** machine, we have the following fragments:

```
...  
VARIABLES  
  accounts , balance  
INVARIANT  
  accounts  $\subseteq$  ACCOUNT  $\wedge$   
  balance  $\in$  accounts  $\rightarrow$   $\mathbb{N}$   
INITIALISATION  
  accounts , balance := { } , { }  
  ...  
OPERATIONS  
  Withdraw ( amount , account )  $\hat{=}$   
    PRE amount  $\in$   $\mathbb{N}$   $\wedge$  account  $\in$  accounts  $\wedge$  amount  $\leq$  balance ( account )  
    THEN  
      balance := balance  $\Leftarrow$  { account  $\mapsto$  balance ( account ) - amount }  
    END ;
```

Part of one of the proof obligations is to establish

$$balance(account) - amount \in \mathbb{N}$$

Prove this.

(12 marks)

5. The following machines are taken from the B demo machines, and describe a person data base. Non-relevant details have been omitted. Read through this listing, then answer the questions on the following page.

MACHINE *data_base_context*

SETS

$SEX = \{ man, woman \};$
 $STATUS = \{ living, dead \}$

END

MACHINE *person_data_base* (*maxpers*)

SEES

data_base_context , *Bool_TYPE*

VARIABLES

person , *sex* , *status* , *mother* , *husband*

INVARIANT

$person \in 0 .. maxpers \wedge$
 $sex \in 1 .. person \rightarrow SEX \wedge$
 $status \in 1 .. person \rightarrow STATUS \wedge$
 $mother \in 1 .. person \leftrightarrow \text{dom} (husband) \wedge$
 $husband \in sex^{-1} [\{ woman \}] \leftrightarrow sex^{-1} [\{ man \}]$

INITIALISATION

$person, sex, status, mother, husband := 0, \{\}, \{\}, \{\}, \{\}$

OPERATIONS

mod_mother (*xx* , *yy*) $\hat{=}$
PRE
 $xx \in 1 .. person \wedge yy \in \text{dom} (husband)$
THEN
 $mother (xx) := yy$
END ;

...

DEFINITIONS

$wife \hat{=} husband^{-1}$

END

(a) Why is *data_base_context* a separate machine?

(3 marks)

(b) Explain the meaning of the last two invariant clauses (those that mention *husband*).

(3 marks)

(c) Derive the proof obligations for **mod_mother**.

(3 marks)

(d) Write an operation **is_sibling**(*xx, yy*) that returns TRUE if *xx* and *yy* are siblings (brothers or sisters), and FALSE otherwise. (You may assume the operation `bool` that converts a predicate to a Boolean value.)

(3 marks)

6. The following code is taken from the complete Library machine discussed in lectures, and relates to the operation of reserving a book. Many non-relevant details have been omitted. Read through this listing, then answer the questions on the following page.

INVARIANT

$$\begin{aligned}
& \text{books_on_loan} \in \text{books_in_library} \leftrightarrow \text{users} \wedge \\
& \text{dom} (\text{books_on_loan}) \cap \text{books_on_shelf} = \{ \} \\
& \text{reserved} \in \text{books_in_library} \leftrightarrow \text{iseq} (\text{users}) \wedge \\
& \text{reserved} \in \text{books_in_library} \leftrightarrow \text{seq}_1 (\text{users}) \wedge \\
& \forall \text{book} . (\text{book} \in \text{dom} (\text{reserved}) \Rightarrow \\
& \quad \text{size} (\text{reserved} (\text{book})) \leq \text{maxreserve}) \wedge \\
& \text{dom} (\text{reserved}) \subseteq \text{dom} (\text{books_on_loan}) \cup \text{dom} (\text{collect}) \wedge \\
& \text{collect} \in \text{books_in_library} \leftrightarrow \text{users} \wedge \\
& \text{dom} (\text{collect}) \cap \text{dom} (\text{books_on_loan}) = \{ \} \wedge \\
& \text{dom} (\text{collect}) \cap \text{books_on_shelf} = \{ \}
\end{aligned}$$

OPERATIONS

$$\begin{aligned}
& \text{Reserve} (\text{user} , \text{book}) \hat{=} \\
& \quad \text{PRE } \text{user} \in \text{users} \wedge \text{book} \in \text{books_in_library} \wedge \\
& \quad \quad \text{book} \in \text{dom} (\text{books_on_loan}) \cup \text{dom} (\text{collect}) \wedge \\
& \quad \quad (\text{book} \in \text{dom} (\text{books_on_loan}) \Rightarrow \\
& \quad \quad \quad \text{books_on_loan} (\text{book}) \neq \text{user}) \wedge \\
& \quad \quad (\text{book} \in \text{dom} (\text{collect}) \Rightarrow \text{collect} (\text{book}) \neq \text{user}) \wedge \\
& \quad \quad (\text{book} \in \text{dom} (\text{reserved}) \Rightarrow \\
& \quad \quad \quad \text{size} (\text{reserved} (\text{book})) \neq \text{maxreserve}) \wedge \\
& \quad \quad (\text{book} \in \text{dom} (\text{reserved}) \Rightarrow \\
& \quad \quad \quad \text{user} \notin \text{ran} (\text{reserved} (\text{book}))) \\
& \quad \text{THEN IF } \text{book} \notin \text{dom} (\text{reserved}) \\
& \quad \quad \text{THEN } \text{reserved} (\text{book}) := [\text{user}] \\
& \quad \quad \text{ELSE } \text{reserved} (\text{book}) := \text{reserved} (\text{book}) \leftarrow \text{user} \\
& \quad \text{END} \\
& \text{END ;}
\end{aligned}$$

(a) Why is the intersection between $\text{dom}(\text{books_on_loan})$ and books_on_shelf empty?

(2 marks)

(b) Why is *reserved* declared as *both* an injective sequence, and a non-empty sequence?

(3 marks)

(c) Explain each of the conjuncts of the **Reserve** operation precondition in prose form (short dot points will suffice).

(7 marks)

7. Perform the following predicate transformations and simplify the resultant predicate:

(a) $[x := new, y := \{new, old\}]p \subseteq books - \{new, x\} \cup y$

(3 marks)

(b) $[p \in books \Rightarrow holdings := holdings \cup \{p\}] \forall z.(z \in books \Rightarrow z \in holdings)$

(3 marks)

(c) Show that both of the sequences $y' := x + y; x' := y - x; y' := y' - y$ and $t := x; x' := y; y' := t$ establish $x' = y \wedge y' = x$, that is, they swap the values of x and y

(6 marks)

8. The following machine defines a set of natural numbers. The machine has just two operations, to add and remove a natural number from the represented set.

MACHINE *NatSet* (*maxnat*)

VARIABLES

natset

INVARIANT

$natset \subseteq 0 .. maxnat$

INITIALISATION

$natset := \{\}$

OPERATIONS

add (*nn*) $\hat{=}$

PRE $nn \in 0 .. maxnat$

THEN $natset := natset \cup \{ nn \}$

END ;

remove (*nn*) $\hat{=}$

PRE $nn \in 0 .. maxnat$

THEN $natset := natset - \{ nn \}$

END

val \leftarrow **cardinality** $\hat{=}$

BEGIN $val := card (natset)$

END

END

The following machine refines the above machine, and performs a data refinement by representing the set of natural numbers in a sequence. For example, if *NatSet* stores the set $\{13, 18, 56\}$, then one possible state of the refinement *NatSetR* is $\langle 18, 56, 13 \rangle$. (Others may be given by a different order of adding the elements, for example $\langle 18, 13, 56 \rangle$ is another possible representation of the original machine state.)

In the spaces provided,

- (a) Define the refinement relation that constrains the state of the refining machine to be consistent with that of the refined (original) machine.
- (b) Define the operations *add*, *remove* and *cardinality* of the refining machine.

REFINEMENT *NatSetR*

REFINES *NatSet*

VARIABLES

setvals

INVARIANT

$setvals \in \text{iseq}(\mathbb{N}) \wedge$

INITIALISATION

$setvals := []$

OPERATIONS

add (*nn*) $\hat{=}$

;

remove (*nn*) $\hat{=}$

;

val \leftarrow **cardinality** $\hat{=}$

END

(12 marks)

9. The following tutorial example defines the *BagMath* machine. Explain i) the definition *Bag*, and ii) the use of the 3 constants *emptyBag*, *BagCount*, *BagUnion*. You may assume that the machine *Value_TYPE* defines the set *VALUE*.

MACHINE *BagMath*

SEES *Value_TYPE*

CONSTANTS

emptyBag , *BagCount* , *BagUnion*

PROPERTIES

$emptyBag \in VALUE \rightarrow \{ 0 \} \wedge$

$BagCount \in Bag (VALUE) \rightarrow (VALUE \rightarrow \mathbb{N}) \wedge$

$\forall bb . (bb \in Bag (VALUE) \Rightarrow BagCount (bb) = emptyBag \triangleleft bb) \wedge$

$BagUnion \in Bag (VALUE) \times Bag (VALUE) \rightarrow Bag (VALUE) \wedge$

$\forall (b1 , b2) . (b1 \in Bag (VALUE) \wedge b2 \in Bag (VALUE) \Rightarrow$

$BagUnion (b1 , b2) = \{ vv , nn \mid vv \in dom (b1) \cup dom (b2) \wedge$

$(nn \in \mathbb{N}_1 \wedge nn = BagCount (b1) (vv) + BagCount (b2) (vv)) \}$

DEFINITIONS

$Bag (X) \hat{=} X \leftrightarrow \mathbb{N}_1$

END

(12 marks)

10. In an example discussed in lectures, the specification of a *SimpleLibrary* machine contained the operation

```

MACHINE SimpleLibrary ( BOOK , maxuser )
...
OPERATIONS Borrow ( user , book )  $\hat{=}$ 
    PRE user  $\in$  users  $\wedge$  book  $\in$  books_on_shelf
    THEN books_on_shelf := books_on_shelf - { book } ||
        books_on_loan := books_on_loan  $\cup$  { book  $\mapsto$  user }
    END

```

The following B fragments define a refinement to *SimpleLibrary*

```

MACHINE SimpleLibraryAPI ( BOOK , maxuser )
    CONSTRAINTS maxuser  $\in$   $\mathbb{N}1$ 
    INCLUDES SimpleLibrary ( BOOK , maxuser )
    SETS
        RESPONSE = { OK , BookInLibrary , NoNewUsers , NotRegisteredUser ,
            BookNotForLoan , BookNotOnLoan }
    OPERATIONS
    ...
    END

```

Define a **robust** operation $response \leftarrow BorrowR(book)$ that has trivial preconditions, and borrows a book *book* for user *user* from the library with response *OK* if it is available to be borrowed, and otherwise returns an appropriate error response. (*You may continue your answer overleaf*)

(Question 10 Answer continued)

(12 marks)

END OF EXAMINATION QUESTIONS