

CSE4213 Examination
2004
Instructions to Candidates

1. Examination time 2 hours
2. Answer all questions
3. There are 10 questions
4. Each question is worth 8 marks
5. Total marks 80
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 4 attributes concerning non-determinism and its role in the B Method.

(a)

(2 marks)

(b)

(2 marks)

(c)

(2 marks)

(d)

(2 marks)

2. You are a recent graduate, and are now working for a large software development company. The company announces that it has been awarded a contract for a safety critical system. You remember that you learnt about how the B Method has been used to build the signalling system for the driverless trains in the Paris Metro. But nobody in the software company has heard about the B Method. What do you tell your colleagues about the B Method? (Remember that they are busy people, and will not listen to lengthy explanations. That is, keep your answer brief: dot points will suffice.)

(8 marks)

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

(2 marks)

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

(2 marks)

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

(2 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.

(2 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

$balance := balance \ominus \{ account \mapsto balance (account) - amount \}$

END ;

Part of one of the proof obligations is to establish

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

Prove this.

(8 marks)

5. A common form of proof obligation arises from function updating. If f is a function $f \in s \rightarrow t$ then a typical update substitution might be

$$f := f \triangleleft \{a \mapsto b\}$$

This gives a proof obligation of the form

$$f \triangleleft \{a \mapsto b\} \in s \rightarrow t$$

- (a) Explain the use of rewriting rules in theorem proving

(3 marks)

(b) Using the rewrite rules

$$f \in u \rightarrow t \quad (1)$$

$$u \cup \{a\} = s \quad (2)$$

$$b \in t \quad (3)$$

break the proof obligation down into three subgoals for the update of

$$\textit{balance} := \textit{balance} \Leftarrow \{\textit{acc} \mapsto 0\}$$

where the invariant states that $\textit{balance} \in \textit{accounts} \rightarrow \mathbb{N}$

(4 marks)

(c) Explain why the variable *accounts* must be treated as a special case.

(1 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

```

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

```

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

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

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

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

(a) $[x := 5 - y]x - 5 > 0$

(2 marks)

(b) $[car, carsInPark := newcar, carsInPark - \{car\}]card(carsInPark \cup \{newcar\}) < capacity$
(2 marks)

(c) $[dir \in \{up\} \Rightarrow mov := mov \triangleleft \{lft \mapsto dir\}] \exists l. (l \in LIFT \Rightarrow mov(l) = up)$

(2 marks)

(d) $[y := x + 1; z := y - 1] x < z + y$

(2 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

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 two operations *add* and *remove* 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*) $\hat{=}$

END

9. Consider the following data model specification for a simple *String* machine. The machine *SEES* the type machine *CHAR_Type*, which defines the set *CHAR*. *strs* is the set of all currently defined strings, and *values* defines the string value of each of the defined strings.

In the space provided, complete the definition of the *Substring* operation, which returns a new string given by the substring starting at *start* and finishing at *end* (inclusive, 1-origin) within the given string *str1*. (Hint: A new string is an element of the set *STRINGS* that is not in the current set of defined strings *strs*.)

(8 marks)

MACHINE *String*

SEES

CHAR_Type

SETS

STRINGS

VARIABLES

strs , *values*

INVARIANT

$strs \subseteq STRINGS \wedge$

$values \in strs \rightarrow seq(CHAR)$

INITIALISATION

$strs, values := \{\}, \{\}$

OPERATIONS

$str \leftarrow \mathbf{Substring} (str1 , start , end) \hat{=}$

END

10. (a) One of the rules for establishing loop correctness is the **The I Rule**: The initialisation substitution establishes the invariant. State the other four of the five rules for loop correctness.

i.

(1 marks)

ii.

(1 marks)

iii.

(1 marks)

iv.

(1 marks)

- (b) Evaluate these four loop correctness rules for the following loop to show that they all are true, and the loop is therefore properly formed.

```
[ xx := 10 ;  
  WHILE xx > 0 DO  
    xx := xx - 1 VARIANT xx INVARIANT xx : NAT  
  END ] xx = 0
```

- i. The I rule is obviously true
ii.

(1 marks)

iii.

(1 marks)

iv.

(1 marks)

v.

(1 marks)

END OF EXAMINATION QUESTIONS