

**CSE4213 Examination
2003
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. State 4 reasons as to why formality is important to software specification.

(8 marks)

2. Write a brief comment (no more than 100 words) on the role of discharging proof obligations in B.

(8 marks)

3. 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.

(8 marks)

4. The following B specification has been discussed in lectures. Read through this specification, then answer the questions following the specification (overleaf).

```

MACHINE SimpleLibrary ( BOOK , maxuser )
CONSTRAINTS maxuser  $\in \mathbb{N}_1$ 
SETS USER
PROPERTIES  $\text{card} ( \textit{USER} ) = \textit{maxuser}$ 
VARIABLES users , books_in_library , books_on_shelf , books_on_loan
INVARIANT
  users  $\subseteq \textit{USER} \wedge$ 
  books_in_library  $\subseteq \textit{BOOK} \wedge$ 
  books_on_shelf  $\subseteq \textit{books\_in\_library} \wedge$ 
  books_on_loan  $\in \textit{books\_in\_library} \leftrightarrow \textit{users} \wedge$ 
  books_on_shelf = books_in_library -  $\text{dom} ( \textit{books\_on\_loan} )$ 
INITIALISATION
  users ,
  books_in_library ,
  books_on_shelf ,
  books_on_loan := { } , { } , { } , { }
OPERATIONS
  AddBook ( book )  $\hat{=}$ 
    PRE book  $\in \textit{BOOK} \wedge \textit{book} \notin \textit{books\_in\_library}$ 
    THEN books_in_library := books_in_library  $\cup \{ \textit{book} \}$  ||
      books_on_shelf := books_on_shelf  $\cup \{ \textit{book} \}$ 
    END ;
  newuser  $\leftarrow$  NewUser  $\hat{=}$ 
    PRE users  $\neq \textit{USER}$ 
    THEN
      ANY user
      WHERE user  $\in \textit{USER} - \textit{users}$ 
      THEN users := users  $\cup \{ \textit{user} \}$  ||
        newuser := user
      END
    END ;
  Borrow ( user , book )  $\hat{=}$ 
    PRE user  $\in \textit{users} \wedge \textit{book} \in \textit{books\_on\_shelf}$ 
    THEN books_on_shelf := books_on_shelf - { book } ||
      books_on_loan := books_on_loan  $\cup \{ \textit{book} \mapsto \textit{user} \}$ 
    END ;
  Return ( book )  $\hat{=}$ 
    PRE book  $\in \text{dom} ( \textit{books\_on\_loan} )$ 
    THEN books_on_shelf := books_on_shelf  $\cup \{ \textit{book} \}$  ||
      books_on_loan := { book }  $\Leftarrow \textit{books\_on\_loan}$ 
    END ;
  user  $\leftarrow$  Borrowed ( book )  $\hat{=}$ 
    PRE book  $\in \text{dom} ( \textit{books\_on\_loan} )$ 
    THEN user := books_on_loan ( book )
    END
END

```

- (a) Explain why *BOOK* is a parameter of the machine, while *USER* is local to the machine.

(1 marks)

- (b) Why is *books_on_loan* defined as a partial function, rather than a total function?

(1 marks)

(c) Give a formal statement of the proof obligation for the initialisation.

(3 marks)

(d) The second substitution in the operation *Borrow* could have been written in function update form as $books_on_loan(book) := \{book \mapsto user\}$. Give the variable update form of this ($books_on_loan := \dots$), and explain how this is different to the use of the union operator. Why is the union valid in this case?

(3 marks)

5. Explain how the mathematical model of a function as used in B can be used to model both structured data values (as in the C data structure `struct`) and object based systems (as in Object Oriented languages). (Hint: Recall the list machine discussed in tutorials and the file machine discussed in lectures.)

(8 marks)

6. A television station is planning to use B to specify its programming (in the non-computing sense). A day's program consists of a sequence of shows (such as *Neighbours*, *Big Brother* and *News*), each of some fixed duration (such as, but not limited to, 30/45/60 minutes).

(a) In the specification, there are two variables: *duration* which defines the duration in minutes of any given item which is an element of the set *SHOWS*, and *program* which defines the station's program for the day. Give typing predicates for the variables (fill in the gap after the \in symbols):

duration \in

program \in

(2 marks)

(b) A refinement of this machine defines a new variable *whatson* with type $\text{whatson} \in 0..1440 \rightarrow \text{SHOWS}$, which maps an arbitrary time of day (there are 1440 minutes in a day) to the current item being transmitted by the station. Give the refinement relation between this variable and the base variables.

(5 marks)

(c) A television program published in a newspaper gives the program for each of a number of channels, specifically the channels 2,7,9,10,28. Give a generalisation of the *whatson* function that captures this fact.

(1 marks)

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

(8 marks)

(a) $[x := x - 1]x < y - 1$

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

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

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

8. Consider the following data model specification for the simple list machine discussed in tutorials. In the space provided, complete the definition of the *cons* operation, which adds an element containing the value *new* to the front of the list.

MACHINE *SimpleList* (*TYPE*)

SETS

POINTERS

CONSTANTS

nil

PROPERTIES

$nil \in POINTERS$

VARIABLES

next , *val* , *list* , *head* , *curpos*

INVARIANT

$list \subseteq POINTERS \wedge$

$next \in POINTERS \rightsquigarrow POINTERS \wedge$

$val \in POINTERS \rightarrow TYPE \wedge$

$head \in list \wedge curpos \in list$

INITIALISATION

$list$, $next$, val , $head$, $curpos := \{nil\}$, $\{\}$, $\{\}$, nil , nil

OPERATIONS

$cons (new) \hat{=}$

END

(8 marks)

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

```

MACHINE SimpleLibrary ( BOOK , maxuser )
...
SETS USER
...
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 ;

...
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 $response \leftarrow BorrowR(user, book)$ that has a trivial precondition. If the book *book* is available for loan, it should record the borrowing by registered borrower *user* and return a response of *OK*. Otherwise it should return an appropriate response.

(8 marks)

(You may continue your answer overleaf)

(Question 9 Answer continued)

10. (a) What are the five rules for establishing loop correctness?

i.

ii.

iii.

iv.

v.

(5 marks)

(b) Verify **any three** of the loop correctness rules for the following loop.

```
[ xx :- 0 ;  
  WHILE xx < 10 DO  
    xx := xx + 1  
    VARIANT 10-xx  
    INVARIANT xx : NAT & xx <= 10  
  END ]  
xx = 10  
i.
```

ii.

iii.

(3 marks)