

**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 about the B Method that justify its role in software development.

(a)

*(2 marks)*

(b)

*(2 marks)*

(c)

*(2 marks)*

(d)

*(2 marks)*

2. Imagine that you are a consulting software engineer. A client approaches you to build a large software system. The system is not safety critical, but there will be significant costs if the delivered system does not perform according to specification. When you explain that you will use the B Method to develop his software, he becomes agitated, and states that he cannot afford to wait too long for the system, and wants you to start "developing code immediately", and that there is an urgency to testing the system in time for delivery. What do you say to him? (Remember that he is a busy person, 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 the invariant in a B specification. (2 marks)

(b) Explain the role of substitutions in a B specification. (2 marks)

(c) In the Generalized Substitution Language of the B Method, the form  $[G]R$  is called a *predicate transformation*. Explain why, and give a simple example. (2 marks)

(d) Give the predicate transform representation of the proof obligation for the operation  
WhatIsThis  $\hat{=}$  **PRE P THEN G END** (2 marks)

4. (a) Explain the difference between a *precondition* and a *guard*, and give an example of each.  
(2 marks)

(b) Explain the term *weakest precondition*. Why is it important?  
(2 marks)

(c) Explain how in the traffic light specification discussed in lectures and tutorials, preconditions ensure both safety and sequencing constraints.  
(2 marks)

(d) A pedestrian light specification is to be added to the simple two way traffic light system of part (c) which INCLUDES the basic two way system. What constraints exist in the including machine (the pedestrian crossing machine) on the INVARIANT of the included machine (the two way traffic lights machine)?  
(2 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

$$f \in u \rightarrow t \tag{1}$$

$$u \cup \{a\} = s \tag{2}$$

$$b \in t \tag{3}$$

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

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

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

(4 marks)

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

(1 marks)

6. 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?

(2 marks)

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

(2 marks)

(c) Derive the proof obligations for **mod\_mother**.

(2 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.)

(2 marks)

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

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

(2 marks)

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

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

(2 marks)

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

(2 marks)

8. A specification uses a data structure of a set *COLOUR* defined as

**SETS**

*COLOUR*

**PROPERTIES**

$COLOUR = \{ red, green, blue \}$

**OPERATIONS**

$add\_col(cc) \hat{=} \mathbf{PRE} \ cc \in COLOUR$

$\mathbf{THEN} \ col := col \cup \{ cc \}$

**END**

Other operations such as *subtract\_col* are similarly defined.

The specification is refined, and the data structure is refined to an array of three booleans. “Boolean” (0 or 1) is used in the C sense of boolean, where a 1 represents inclusion of the corresponding element in the set. The refinement uses the library machine *Renaming\_Narr* in the following way:

**CONSTANTS**

*red\_R, green\_R, blue\_R, map*

**PROPERTIES**

$red\_R = 0 \wedge green\_R = 1 \wedge blue\_R = 2 \wedge$

$map \in \mathbb{N} \rightarrow COLOUR \wedge$

$map = \{ red\_R \mapsto red, green\_R \mapsto green, blue\_R \mapsto blue \}$

**OPERATIONS**

$add\_col\_R(cc) \hat{=} \mathbf{PRE} \ cc \in \{ red\_R, green\_R, blue\_R \}$

$\mathbf{THEN} \ col\_Narr\_STO(cc,1)$

**END**

Define the refinement invariant between *col* and *col\_R*.

(8 marks)

9. Consider the following data model specification for a simple list machine. In the space provided, complete the definition of the *append* operation, which adds a new element containing the value *new* to the end of the list.

(8 marks)

**MACHINE** *SimpleList* ( *TYPE* )

**SETS**

*POINTERS*

**CONSTANTS**

*nil*

**PROPERTIES**

$nil \in POINTERS$

**VARIABLES**

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

**INVARIANT**

$list \subseteq POINTERS \wedge$

$next \in POINTERS \rightsquigarrow POINTERS \wedge$

$val \in POINTERS \leftrightarrow TYPE \wedge$

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

**INITIALISATION**

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

**OPERATIONS**

$append ( new ) \hat{=}$

**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 one of them fails, and the loop is therefore not properly formed.

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

i. The I rule is obviously true

ii.

*(1 marks)*

iii.

*(1 marks)*

iv.

*(1 marks)*

v.

*(1 marks)*

**END OF EXAMINATION QUESTIONS**