

**CSE4213 Examination
2002
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 advantages of using the B Method for software development

(8 marks)

- (a) **Answer:** Covers the complete life cycle from specification to maintenance (including refinement, implementation, code generation)
- (b) **Answer:** Generated code can be proven to be consistent with specification
- (c) **Answer:** Emphasis is upon the **what** and not the **how**
- (d) **Answer:** Better suited to discrete domains, rather than analogue
- (e) **Answer:** Discharging proof obligations is the counterpart of testing

Answer: Two marks for each correct answer

Comment: Very straightforward: few students had any difficulty with this question.

2. Write a brief comment (no more than 100 words) on the role of formality in software engineering.

(8 marks)

Answer: The progress of software engineering has been characterised by an increase in formality. Early programming language development was greatly enhanced by the use of formal grammars. Programming languages themselves have evolved through the application of increased formality to their design, specification and implementation; for example through the use of formal semantics and attribute grammars. The software development process has been improved by the development of more formal design methods. Everywhere that formality has been used, there is an attendant increase in reliability and correctness.

Comment: This question was based upon one particular slide and several comments in one lecture, but some students had obviously not taken this in! The emphasis was upon the historical evolution of formality, but other aspects could get full marks if adequately argued.

3. Explain the difference between **specification** and **implementation** in the light of formal methods such as the B Method. (Hint: you may wish to illustrate your answer by referring to the Y2K bug.)

(8 marks)

Answer: Specification deals with the abstract constructs; Implementation deals with the finite constructs. The Y2K bug was caused by the implementation (using 2 digits for space reasons) being confused with the specification (requiring an unbounded (not 4) set of digits) to represent the year number.

Comment: The **how** and the **what** were key words in many students answers, and were rewarded as were the key words **finite/unbounded**, **abstract/concrete**, and the like. Few problems in general.

4. In the **Square** machine, we formed an approximate square root of a given number by using a non-deterministic approximation *approx*, given by

```
sqrt <-- ApproxSqrt(num) =
  PRE num : NAT
  THEN ANY approx
    WHERE approx : NAT &
      approx*approx <= num &
      num < (approx+1)*(approx+1)
    THEN sqrt := approx
  END
END
```

Explain why this use of non-determinism is appropriate, and what must be done in refining this specification towards an implementation.

(8 marks)

Answer: Non-determinism is very useful in specification, as it allows us to focus upon the **what**, rather than the **how**. It abstracts over the actual computation of *approx*, and instead specifies what properties it must have. In building an implementation from this, we must develop an algorithm to compute the value satisfying the specified properties, where the algorithm can be regarded as a stronger statement of the specification, but provably consistent.

Comment: This question started to separate the levels of ability. The point many students omitted was the need to move the specification in the direction of developing an **algorithm** to compute (rather than define) the required value.

5. Write a fragment of a B machine specification to show the use of a *deferred* set. Explain the difference between a *deferred set* and a *set parameter*.

(8 marks)

Answer:

```
MACHINE Deferred
SETS USER
...
```

The set *USER* is a deferred set. It differs from a set parameter in that although it is not known what value it has until the machine is instantiated (like a parameter), it is a component of the machine itself, rather than being some externally supplied component.

Comment: Really bookwork, but many students did not know the difference between deferred and parameter sets. There were 4 marks attached to reproducing the above B fragment: really a giveaway! The other 4 marks were given to those students who identified (and explained) that parameter sets were **externally** defined, while deferred sets were **internal** to the machine being specified.

6. If we were modelling a taxi fleet company we might have three variables, *drivers*, *taxis* and *assigned* constrained by

$$\begin{aligned} \textit{drivers} &\in \mathbb{P} \text{ DRIVERS} \\ \textit{taxis} &\in \mathbb{P} \text{ TAXIS} \\ \textit{assigned} &\in \textit{drivers} \succ\!\!\rightarrow \textit{taxis} \end{aligned}$$

where *drivers* is the set of drivers working for the company, *taxis* is the set of taxis owned by the company, and *assigned* is a function recording the assignment of drivers to taxis.

The arrow $\succ\!\!\rightarrow$ denotes a *partial injective* function.

- (a) Describe in your own words what a *partial injective* function is (that is, do **not** give a formal description).

(4 marks)

Answer: A function is a mapping from a set S to a set T , where each value in S can have at most one mapping. It is a special case of a *relation*. *partial* means that not all values in the set S are in the *domain* of the function. *injective* means that no two different values in S map to the same value in T .

- (b) Why is *assigned* a partial function?

(1 marks)

Answer: Because not all drivers may be assigned a taxi

- (c) Why is *assigned* an injective function?

(1 marks)

Answer: Because each taxi can be assigned to only one driver

- (d) Specify the drivers who are currently assigned.

(1 marks)

Answer: $\text{dom}(\textit{assigned})$

- (e) Specify the taxis that are currently unassigned.

(1 marks)

Answer: $\textit{taxis} - \text{ran}(\textit{assigned})$

Comment: Key words in collecting the 4 marks for part a are those italicised words in the answer. It was inappropriate to mention taxis at this point. Parts b and c were looking for insights into the partialness and injectiveness respectively: confusing the two aspects penalised some. d and e caused little difficulty, although I was a little surprised to see some students use more elaborate means of specification (such as set comprehension).

7. Perform the following predicate transformations and simplify the resultant predicate:
(8 marks)

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

Answer: $x < y$

(b) $[x := a + b, y := c + d] \exists p. p = \frac{x}{y} \wedge p < 100$

Answer: $\exists p. p = \frac{a+b}{c+d} \wedge p < 100$, which can be simplified to $\frac{a+b}{c+d} < 100$

(c) $[x > y \Rightarrow x := x - 1]y \leq x$

Answer: $x > y \Rightarrow y \leq x - 1$

(d) **[CHOICE** $p := new$ **OR** $users := users - \{p\}$ **END]** $p \notin users$

Answer: $[p := new] p \notin users \wedge [users := users - \{p\}] p \notin users$

which simplifies further to $new \notin users \wedge p \notin users - \{p\}$

As the right hand conjunction is a tautology (always true), this further simplifies to $new \notin users$

Comment: Parts a and b caused little difficulty. I did expect people to simplify $x + 1 < y + 1$, but did not penalise those who failed to remove the existential quantifier in b.

For some reason, many students left off the LHS of the implication in c. While it is true for natural numbers that c is a tautology, you cannot simplify this further without additional constraints.

Either of the expressions $new \notin users \wedge p \notin users - \{p\}$ or $new \notin users$ collected two marks (the emphasis was on the substitutions, not the simplifications).

8. Compute the proof obligations for the operation **ToGreen** in the following specification:

MACHINE *SimpleTwoWay*

SETS

$DIRECTION = \{ NorthSouth, EastWest \};$

$LIGHT = \{ Red, Green, Amber \}$

VARIABLES

lights

INVARIANT

$lights \in DIRECTION \rightarrow LIGHT \wedge$

$(lights (NorthSouth) \in \{ Green, Amber \} \Rightarrow lights (EastWest) = Red) \wedge$

$(lights (EastWest) \in \{ Green, Amber \} \Rightarrow lights (NorthSouth) = Red)$

INITIALISATION

$lights := \{ NorthSouth \mapsto Red, EastWest \mapsto Red \}$

OPERATIONS

ToRed (*dir*) $\hat{=}$

PRE $dir \in DIRECTION \wedge lights (dir) = Amber$

THEN $lights (dir) := Red$

END ;

ToGreen (*dir*) $\hat{=}$

PRE $dir \in DIRECTION \wedge lights (dir) = Red \wedge$

$(dir = NorthSouth \Rightarrow lights (EastWest) = Red) \wedge$

$(dir = EastWest \Rightarrow lights (NorthSouth) = Red)$

THEN $lights (dir) := Green$

END ;

ToAmber (*dir*) $\hat{=}$

PRE $dir \in DIRECTION \wedge lights (dir) = Green$

THEN $lights (dir) := Amber$

END

END

(8 marks)

(Write your answer overleaf)

(Question 8 Answer Page)

Answer: Use the rule

$$I \wedge P \Rightarrow [G]I$$

Expanding on G, the substitution for lights, we have:

$$\begin{aligned} & [\text{lights} (\text{dir}) := \text{Green}] \\ & = [\text{lights} := \text{lights} \triangleleft \{ \text{dir} \mapsto \text{Green} \}] \end{aligned}$$

so the proof obligation is:

$$\begin{aligned} & \text{lights} \in \text{DIRECTION} \longrightarrow \text{LIGHT} \wedge \\ & (\text{lights} (\text{NorthSouth}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} (\text{EastWest}) = \text{Red}) \wedge \\ & (\text{lights} (\text{EastWest}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} (\text{NorthSouth}) = \text{Red}) \\ & \wedge \text{dir} \in \text{DIRECTION} \wedge \text{lights} (\text{dir}) = \text{Red} \wedge \\ & (\text{dir} = \text{NorthSouth} \Rightarrow \text{lights} (\text{EastWest}) = \text{Red}) \wedge \\ & (\text{dir} = \text{EastWest} \Rightarrow \text{lights} (\text{NorthSouth}) = \text{Red}) \\ & \Rightarrow [\text{lights} := \text{lights} \triangleleft \{ \text{dir} \mapsto \text{Green} \}] \text{lights} \in \text{DIRECTION} \longrightarrow \text{LIGHT} \wedge \\ & (\text{lights} (\text{NorthSouth}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} (\text{EastWest}) = \text{Red}) \wedge \\ & (\text{lights} (\text{EastWest}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} (\text{NorthSouth}) = \text{Red}) \end{aligned}$$

so performing the substitution gives the proof obligation as:

$$\begin{aligned} & \text{lights} \in \text{DIRECTION} \longrightarrow \text{LIGHT} \wedge \\ & (\text{lights} (\text{NorthSouth}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} (\text{EastWest}) = \text{Red}) \wedge \\ & (\text{lights} (\text{EastWest}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \text{lights} (\text{NorthSouth}) = \text{Red}) \\ & \wedge \text{dir} \in \text{DIRECTION} \wedge \text{lights} (\text{dir}) = \text{Red} \wedge \\ & (\text{dir} = \text{NorthSouth} \Rightarrow \text{lights} (\text{EastWest}) = \text{Red}) \wedge \\ & (\text{dir} = \text{EastWest} \Rightarrow \text{lights} (\text{NorthSouth}) = \text{Red}) \\ & \Rightarrow \text{lights} \triangleleft \{ \text{dir} \mapsto \text{Green} \} \in \text{DIRECTION} \longrightarrow \text{LIGHT} \wedge \\ & (\text{lights} \triangleleft \{ \text{dir} \mapsto \text{Green} \} (\text{NorthSouth}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \\ & \quad \text{lights} \triangleleft \{ \text{dir} \mapsto \text{Green} \} (\text{EastWest}) = \text{Red}) \wedge \\ & (\text{lights} \triangleleft \{ \text{dir} \mapsto \text{Green} \} (\text{EastWest}) \in \{ \text{Green} , \text{Amber} \} \Rightarrow \\ & \quad \text{lights} \triangleleft \{ \text{dir} \mapsto \text{Green} \} (\text{NorthSouth}) = \text{Red}) \end{aligned}$$

Comment: After marking a few of the scripts, I changed my mind about the marking scheme, since few people showed working to this detail. Hence I looked for the $I \wedge P$ expansion (4 marks), the implication (1 mark), and the correct substitution (2 marks) and expansion (1 mark) to arrive at the final form shown immediately above. Many students forgot to apply the substitution.

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

```
MACHINE SimpleLibrary ( BOOK , maxuser )  
...  
OPERATIONS  
  AddBook ( book ) =  
    PRE book ∈ BOOK ∧  
        book ∉ books_in_library  
    THEN books_in_library := books_in_library ∪ {book}  
    END ;  
...  
END
```

The following B fragments define a refinement to *SimpleLibrary*

```
MACHINE SimpleLibraryAPI ( BOOK , maxuser )  
CONSTRAINTS maxuser ∈ ℕ1  
INCLUDES SimpleLibrary ( BOOK , maxuser )  
SETS  
  RESPONSE = { OK , BookInLibrary , NoNewUsers , NotRegisteredUser ,  
                BookNotForLoan , BookNotOnLoan }  
OPERATIONS  
...  
END
```

Define a **robust** operation $response \leftarrow AddBookR(book)$ that has a trivial precondition, adds the book *book* to the library with response *OK* if it is not already in *books_in_library*, and otherwise returns the response *BookInLibrary*.

(8 marks)

(You may continue your answer overleaf)

(Question 9 Answer continued)

Answer:

```
MACHINE SimpleLibraryAPI ( BOOK , maxuser )
  CONSTRAINTS maxuser ∈ ℕ1
  INCLUDES SimpleLibrary ( BOOK , maxuser )
  SETS
    RESPONSE = { OK , BookInLibrary , NoNewUsers , NotRegisteredUser ,
      BookNotForLoan , BookNotOnLoan }
  OPERATIONS
    response ← AddBookR ( book ) =
      PRE book ∈ BOOK ♡
      THEN IF book ∉ books_in_library THEN ♡
        AddBook ( book ) || ♡♡♡
        response := OK ♡
      ELSE response := BookInLibrary ♡
      END
    END ;
  END
```

Comment: The marks for each part are shown with ♡ symbols. One mark was lost if the *AddBook* operation was not used, but reimplemented from *SimpleLibrary*. An additional mark was given for any form of reflective comment or explanation. Generally OK, but there were few full marks for this Q.

10. 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

$$\begin{aligned} & \textit{emptyBag} \in \textit{VALUE} \rightarrow \{ 0 \} \wedge \\ & \textit{BagCount} \in \textit{Bag} (\textit{VALUE}) \rightarrow (\textit{VALUE} \rightarrow \mathbb{N}) \wedge \\ & \forall \textit{bb} . (\textit{bb} \in \textit{Bag} (\textit{VALUE}) \Rightarrow \textit{BagCount} (\textit{bb}) = \textit{emptyBag} \triangleleft \textit{bb}) \wedge \\ & \textit{BagUnion} \in \textit{Bag} (\textit{VALUE}) \times \textit{Bag} (\textit{VALUE}) \rightarrow \textit{Bag} (\textit{VALUE}) \wedge \\ & \forall (\textit{b1} , \textit{b2}) . (\textit{b1} \in \textit{Bag} (\textit{VALUE}) \wedge \textit{b2} \in \textit{Bag} (\textit{VALUE}) \Rightarrow \\ & \quad \textit{BagUnion} (\textit{b1} , \textit{b2}) = \{ \textit{vv} , \textit{nn} \mid \textit{vv} \in \textit{dom} (\textit{b1}) \cup \textit{dom} (\textit{b2}) \wedge \\ & \quad \quad (\textit{nn} \in \mathbb{N}_1 \wedge \textit{nn} = \textit{BagCount} (\textit{b1}) (\textit{vv}) + \textit{BagCount} (\textit{b2}) (\textit{vv})) \}) \end{aligned}$$

DEFINITIONS

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

END

(8 marks)

Answer:

emptyBag defines an empty bag as a total function that maps every value of type *VALUE* onto zero, representing the fact that there are zero of every possible values in the bag.

BagCount is a function that takes a bag of *VALUE*s, and returns a function that defines how many instances of each value of type *VALUE* there are in the bag.

BagUnion is a function that takes two bags of identical *VALUE*s, and return a single bag containing the contents of both bags.

Bag is a model of a bag of values, where there may be more than one instance of any particular value. It models the bag as a function that returns the number of instances of any particular value of type *X*.

Comment: Not well done, particularly by those who skipped the last tutorial! Again, the emphasis was upon demonstrating an insight into the use of the notation, rather than just parroting definitions of the notation used.