

**CSE4213 Examination**  
**June 2007**  
**Instructions to Candidates**

1. Examination time 2 hours
2. Answer all questions
3. There are 5 questions
4. Each question is worth 12 marks
5. Total marks 60
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 predicates and their role in the B Method. For each attribute, give a corresponding example that demonstrates the attribute.

(a)

*(2 marks)*

(b)

*(2 marks)*

(c)

*(2 marks)*

- (d) The B Toolkit uses abstract machines in specifying software behaviour. Explain the role of *state* in the use of abstract machines.

*(6 marks)*

2. (a) Explain the role of the *invariant* in B specifications.

(3 marks)

(b) Explain how proof obligations for operations in a B specification are derived.

(3 marks)

(c) Explain why global sets in a B development (involving more than one machine) should not be passed as parameters.

(3 marks)

(d) Explain why guarded substitutions must be used with care.

(3 marks)

3. Define a machine called *ArrayType* to represent an array. The machine should have two parameters, a natural number to fix the upper limit of the indexes (the lower limit is 1) and a set for the type of values to be stored in the array. There should be operations to
- read the value at a given index;
  - store a value at a given index;
  - exchange the values at two indexes.

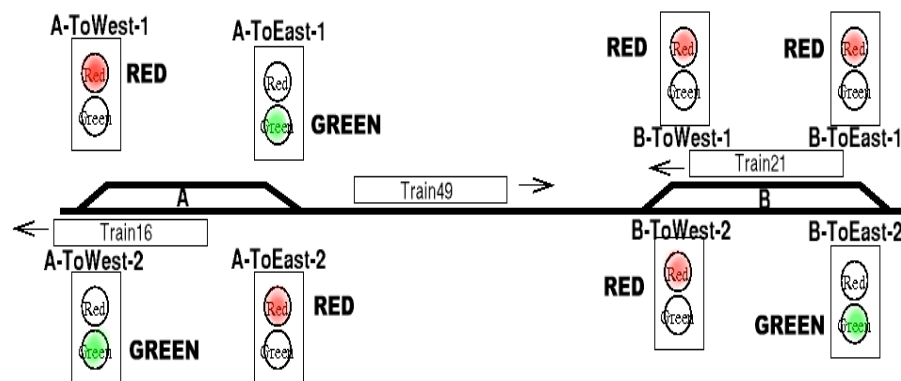
The initialization and the operations to read values from the array and to exchange values in the array should warn the user of the array not to look at values at an index that has not previously had a value stored.

(12 marks)

4. The *TrafficLights* problem discussed in lectures showed how both *safety* and *sequencing* were enforced by the appropriate use of invariant and precondition predicates.

In the following railway scenario, you are to develop similar safety and sequencing rules for the operation of trains on a single piece of track. The constraints are that only one train can be travelling on the section of track at a time, and other trains are not permitted to enter at either end of the track until the occupying train has safely cleared the section. At each end of the section is a passing loop, which allows 2 trains to pass, either in the same direction (overtaking), or the opposite direction.

The following diagram gives an example:



The exit from each track in the loops are controlled by a two-aspect light, red (upper) and green (lower) only, facing towards the center of the loop. The points at each end are assumed to be automatically controlled. The current aspect of each signal is shown alongside the signal.

- Train16 is departing loop A to the west on track 2 (the lower in the diagram), and signal A-ToWest-2 is green.
- Train49 has just crossed with Train16 at Loop A, and has departed from track 1. Signal A-ToEast-1 is still showing green, but is about to change to red. In anticipation of Train49 arriving at Loop B, the signal B-ToEast-2 has changed to green, indicating a clear run through the loop.
- Train21 has been held at Loop B, awaiting the arrival of Train49. Signal B-ToWest-1 is therefore showing red.

Assume the following sets:  $ASPECT = \{Red, Green\}$ ,  $DIR = \{East, West\}$ ,  $TRACK = \{loopAtrack1, loopAtrack2, sectionAB, loopBtrack1, loopBtrack2\}$ .

The machine is to have the following operations:  $ChangeToGreen(signal)$ ,  $ChangeToRed(signal)$ ,  $MoveTrain(trackX, trackY)$ , where the last operation marks a train as moving from  $trackX$  to  $trackY$ .

Answer the following questions on the next page.

(a) Define a suitable state for the machine

*(4 marks)*

(b) Define a suitable invariant for the machine

*(4 marks)*

(c) Define suitable preconditions for the operations.

*(4 marks)*

5. The following code is taken from one of the B demonstration packages for computing the factorial of a number. It is a robust implementation of a specification that computes the factorial if it is representable within the integer range of the computer system, and 0 otherwise.

Explain the key features of the implementation, paying particular attention to the variables *done*, *count*, and *in\_range*. You should make reference to the 5 rules for establishing loop correctness, namely the **Initialisation**, **Final**, **T1 variant natural**, **T2 variant decreasing** and **Preserve invariant** rules.

Useful hints:

**STO\_NVAR** Store a natural number

**SCL** Make argument into a scalar

**EQL** args are equal, TRUE or FALSE

**\_PRE\_MUL\_NVAR** Precondition that result of multiplication is still in range of scalars

**\_MUL\_NVAR** Multiply scalar by the argument

**DEC** decrement a scalar

**math\_fac** the mathematical definition of factorial (unbounded)

## IMPLEMENTATION

This implementation is a robust implementation of factorial

*factorial\_1*

## REFINES

*factorial*

## SEES

*Bool\_TYPE* , *Scalar\_TYPE* , *Scalar\_TYPE\_Ops*

## IMPORTS

An integer machine is used

*factorial\_Nvar* ( *my\_maxint* )

## PROPERTIES

*my\_maxint* = *MaxScalar*

## OPERATIONS

This implementation returns factorial (if possible).

```
ret ← comp_fac ( inp ) ≐
BEGIN
  VAR in_range , done , count IN
    factorial_STO_NVAR ( 1 ) ;
    count ← SCL ( inp ) ;
    done ← EQL ( count , 0 ) ;
    IF done = FALSE THEN
      in_range ← factorial_PRE_MUL_NVAR ( count ) ;
      WHILE in_range = TRUE ∧ done = FALSE DO
        factorial_MUL_NVAR ( count ) ;
        count ← DEC ( count ) ;
        done ← EQL ( count , 0 ) ;
        in_range ← factorial_PRE_MUL_NVAR ( count )
      INVARIANT
        count ∈ ℕ ∧
        count ∈ 0 .. my_maxint ∧
        factorial_Nvar ∈ 0 .. my_maxint ∧
        ( done = TRUE ⇒ count = 0 ) ∧
        ( done = TRUE ⇒ in_range = TRUE ) ∧
        ( done = FALSE ⇒ count ≠ 0 ) ∧
        ( in_range = TRUE ⇒ math_fac ( inp ) = math_fac ( count ) × factorial_Nvar ) ∧
        ( in_range = TRUE ⇒ factorial_Nvar × count ≤ my_maxint )
      VARIANT
        count
      END
    END ;
    IF done = TRUE THEN
      ret ← factorial_VAL_NVAR
    ELSE
      ret := 0
    END
  END
END
END
END
```

Write your answers on the next page

*(12 marks)*

**END OF EXAMINATION QUESTIONS**