UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1997

MEng Honours Degrees in Computing Part IV
MSc Degree in Advanced Computing
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the
Diploma of Membership of Imperial College
Associateship of the City and Guilds of London Institute

PAPER 4.92

THEORIES OF SPECIFICATION AND VERIFICATION

Tuesday, April 29th 1997, 2.00 - 4.00

Answer THREE questions

For admin. only: paper contains 4 questions

1 This question concerns a system for controlling the filling and emptying of a tank of fluid (Figure 1) involving the following components:

1. A push-button switch, which is either on (pressed) or off (released);
2. A fill valve, which can be either open or closed;
3. An exit valve, which is either open or closed;
4. A high-level sensor, which is on if the fluid level is at or above its position, and off otherwise;
5. A low-level sensor, which is on if the fluid level is at or above its position, and off otherwise.

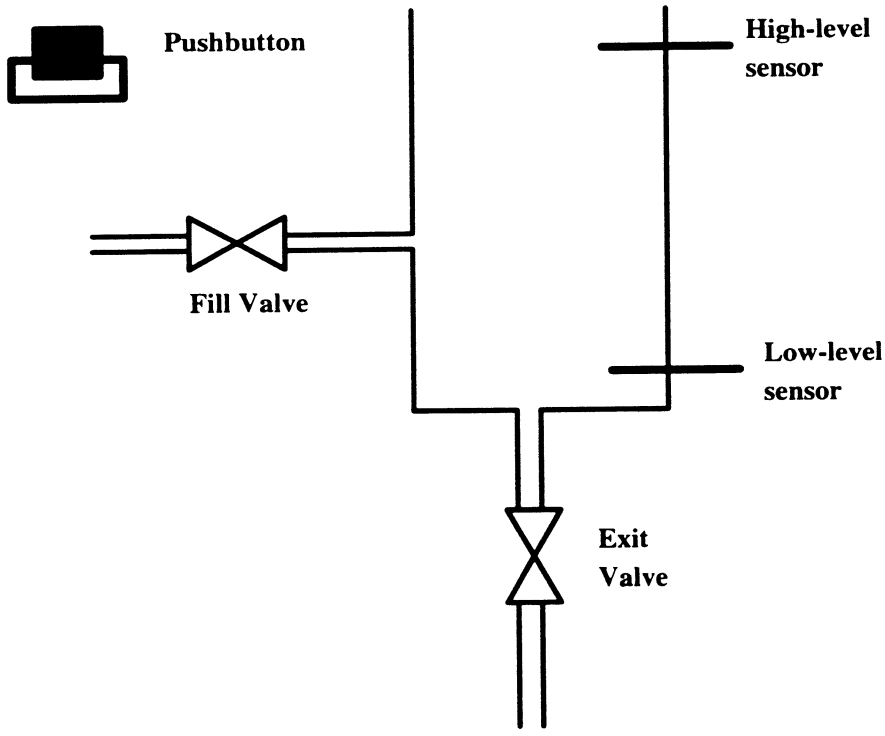

Figure 1: Tank Components

The required functionality of the control system is as follows:

- When the button is pressed, if the valves are closed and both sensors off, then the fill valve should be opened – initiating a "filling" phase

- If the high-level sensor goes on the fill valve should be closed and the exit valve opened – initiating an "emptying" phase
- If the low-level sensor goes off the exit valve should be closed.

The system should also respond to the other sensor events in order to keep an accurate record of their states.

a Formalise the valves as machines **FillValve** and **ExitValve** with single variables which record the valve state, and operations to open and close the valve. Initially both valves can be assumed to be closed.

b Define the state of the **Controller** machine, and use suitable inclusion mechanisms to permit access to the state and operations of the valves. Define variables to record the state of the two sensors – initially these are off.

c Define **Controller** operations to respond to (i) the button press event; (ii) the high-level sensor going on; (iii) the low-level sensor going off; (iv) the high-level sensor going off; (v) the low-level sensor going on.

The three parts carry, respectively, 30%, 30% and 40% of the marks.

Turn over ...

2 This question concerns a munitions storage system. Items are to be stored in storage areas, but all the items in a given area must be able to be safely stored together. There is a relationship "can be stored with" between items, expressed in Figure 2 as a many-many association from an item to the set of items it can be stored with.

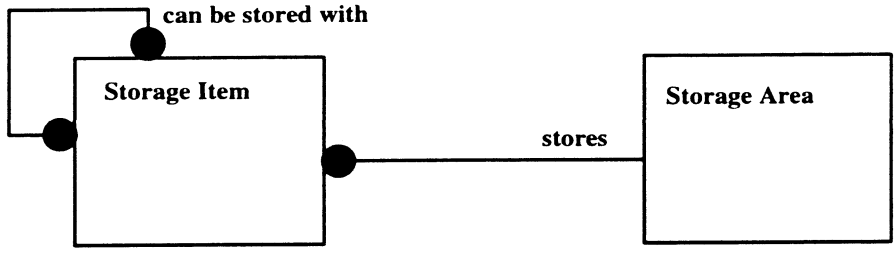

Figure 2: Object Model of Storage System

a Define the data parts of machines **Item** and **Area** to formalise the data model of Figure 2, using a suitable inclusion mechanism between these machines.

b Formalise the invariant of **Item** that if an item **i1** can be stored with another item **i2**, then **i2** can be stored with **i1** (that is, the relationship is symmetric).

c Formalise the invariant of **Area** that all items stored in a given area are related by "can be stored with".

d Define operations to (i) create (a record for) a storage item; (ii) add items to the "can be stored with" set of an item; (iii) create a storage area; (iv) add items to its set of stored items. These operations must maintain the invariants.

The four parts carry, respectively, 30%, 15%, 15% and 40% of the marks.

3 Consider the following machine for the specification of a set of numbers with operations to test membership of a number in the set, and to test if a given number is larger than every element of the set:

```
MACHINE LSet
SEES Bool_TYPE
VARIABLES ss
INVARIANT ss: FIN(NAT)
INITIALISATION ss := {}
OPERATIONS
  add_element(xx) = PRE xx: NAT
                    THEN
                      ss := ss \/ { xx }
                    END;

  bb <-- is_member(xx) =
             PRE xx: NAT
             THEN
               IF xx: ss
               THEN
                 bb := TRUE
               ELSE
                 bb := FALSE
               END
             END;

  bb <-- is_larger(xx) =
             PRE xx: NAT
             THEN
               IF (ss = {}) or (ss /= {} & xx > max(ss))
               THEN
                 bb := TRUE
               ELSE
                 bb := FALSE
               END
             END
END
```

a Write a refinement of this specification, using a sequence as the main data structure. Use loops to implement the enquiry operations, but do not use library machines to implement sequence operations.

b Explain why the refined versions of the operations satisfy their specifications.

The two parts carry, respectively, 75% and 25% of the marks. *Turn over ...*

4

a List the major conceptual differences between B and VDM^{++}

b Consider the munitions storage system data model represented in Figure 3. An **Item** object has a data component **code** which is a sequence of natural numbers, and an operation **set_code**(**x** : $\mathbb{N}$) which is left abstract in the supertype, but is defined differently in the two subtypes: in **HighHazard** it sets **code** to be the sequence [**x**, 2] whilst in **LowHazard** it sets **code** to be [**x**, 1]. Initially the **code** of any item is the empty sequence.

An **Area** object has initially the empty set of contained items, and has an operation to add an item to this set.

Give VDM^{++} specifications of these four classes.

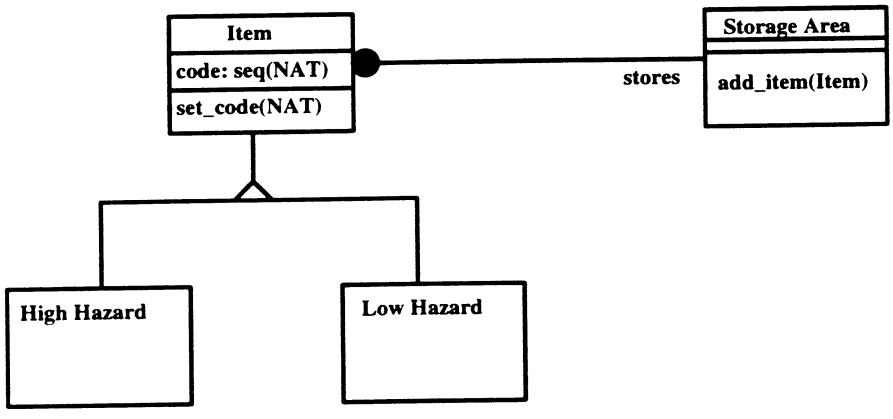

Figure 3: Object Model of Munitions Storage

The two parts carry, respectively, 30% and 70% of the marks.

End of paper.