2CS60

APPROACHES TO ERROR-FREE SOFTWARE

EXAM QUESTIONS - 1999

TWO AND A HALF HOURS

Answer four questions.

If you attempt to answer more than the required number of questions, the mark awarded for the excess question (i.e. the one with the lowest mark) will be discarded.

1 (a). One of the popular misconceptions in the computing community is that formal verification can guarantee that software is perfect.

(i). Why is this a misconception? (4 marks)

(ii). What arguments may be put forward in favour of formal verification nonetheless? (4 marks)

(iii). Explain the difference between the Floyd assertion approach to software verification and Hoare's axiomatic approach. (5 marks)

(b). During the past two decades, much software engineering research and development has involved new notations for formal specification of software systems.

(i). Why is there emphasis on the specification stage in the software development process? (3 marks)

(ii). Why have new notations been developed in preference to existing alternatives? (3 marks)

(iii). Explain the difference between a model-based specification notation and a property-based specification notation. (4 marks)

(iv). What type of approach to specification is the VDM notation and why is it particularly well suited to formal verification of software? (2 marks)

n

2 (a). Explain what is meant by each of the following terms in the context of the VDM specification language, giving a short illustrative example in each case:

(i). map; (4 marks)

(ii). 'seq of X' for some base set X; (4 marks)

(iii). make function. (4 marks)

(b). Consider the following simplified model in the VDM notation for a newsagent's system. Customers are identified by a unique customer number, whose type is Cnum. Information about customers is stored in a composite type CustomerDetails, which has fields to record the customer's name, the customer's address, the daily newspaper ordered by the customer (considered to be valid from Monday to Saturday inclusive) and the Sunday newspaper ordered by the customer, i.e.

```
CustomerDetails :: name    : NameType
                   address : AddressType
                   daily   : DailyPaper
                   sunday  : SundayPaper
```

The state of the system is to be modelled by a customer map:

```
cm : map Cnum to CustomerDetails
```

which relates customer numbers to customer information. The types Cnum, NameType, AddressType, DailyPaper and SundayPaper need not be further defined here; the type N mentioned later refers to the set of non-negative whole numbers.

(i). Give a line-by-line explanation of the following operation specification in VDM and describe its overall effect:
(3 marks)

```
HOW-MANY( p : DailyPaper ) r : N

ext rd cm : map Cnum to CustomerDetails
```

post r= card{ daily(cm(c)) | c dom cm
daily(cm(c)) = p }

(ii). Provide an implicit VDM specification for the following operation:

REMOVE-CUSTOMER(c : Cnum)

This operation removes all trace of the customer with number c from the system.
(2 marks)

Question continues
2 (b).
cont.

(iii). Provide an implicit VDM specification for the following operation:

CHANGE-SUNDAY-PAPER(c : Cnum, p : SundayPaper)

For customer with number c, this operation changes the Sunday paper ordered to become p. (3 marks)

(iv). Provide an implicit VDM specification for the following operation:

PREPARE-DAILY-ROUND(round : seq of Cnum)
pile : seq of DailyPaper

This operation uses round (a list which identifies a particular delivery sequence of unique customer numbers all of which correspond to registered customers), to generate as output, pile, the corresponding ordered list of daily newspapers needed for delivery on the round.
(4 marks)

(v). The model of the newsagent's system as described has a number of restrictions which would need further consideration if a genuine system were to be developed. Outline in brief two of the most significant restrictions.
(1 mark)

n

3 (a). (i). Explain the process of term-rewriting by which expressions are evaluated in the OBJ algebraic specification language. (4 marks)

(ii). Give two formal conditions necessary for evaluation of an expression in OBJ to a unique form, and explain how, in practice, these conditions may be satisfied. (4 marks)

(b). The following OBJ specification is a simplified information system for an estate agent recording a list of houses. Each individual house is represented as H(a,s) where a is the address of the house (of type 'address', a user-defined type), and s is the state of the house (of type 'state', again a user-defined type, and having possible values: forsale, underoffer and unknown).

```
EstateAgent

OBJ House
SORTS address state house
OPS
  H                              : address state -> house
  addr1, addr2, addr3, addr4, addr5 :          -> address
  forsale, underoffer, unknown   :             -> state
JBO

OBJ HouseList / House
SORTS houselist
OPS
  nil    :                -> houselist *** Empty list   ***
  _ & _  : house houselist  -> houselist *** Add house    ***
  remove : address houselist -> houselist *** Remove house ***
VARS
  a?, a  : address
  s      : state
  hlist  : houselist
EQNS
  ( remove(a?, nil)          = nil                    )
  ( remove(a?, H(a,s) & hlist) = hlist          IF a? == a )
  ( remove(a?, H(a,s) & hlist) = H(a,s) & remove(a?, hlist)
                                        IF not a? == a )
```

```
JBO

OBJ Test / HouseList
OPS database : -> houselist
EQNS
  ( database = H(addr1,underoffer) & H(addr2,forsale)
    & H(addr3,forsale) & H(addr4,underoffer) & nil )
JBO.
```

(i). Illustrate the term-rewriting process by giving the steps in the evaluation of the following expression:

remove(addr2,database)

(3 marks)

Question continues

3 (b).
cont.

(ii). Give appropriate equations to incorporate the following operation:

count : houselist -> nat

This operation should deliver a count of the number of houses that are in the forsale state in a given house list, e.g. count(database) should give 2 as its result.

(3 marks)

(iii). Give appropriate equations to incorporate the following operation:

state? : address houselist -> state

This operation should deliver the state of a given address (the first parameter) in a supplied house list (the second parameter), e.g. state?(addr3,database) should give forsale as its result. If the address is not present in the house list, the state unknown should be returned. (3 marks)

(iv). Give appropriate equations to incorporate the following operation:

makeoffer : address houselist -> houselist

This operation should alter the recorded state of the house at the given address (the first parameter) to be underoffer, if it is currently present in the house list (the second parameter) and in the forsale state. In all other circumstances the house list should remain unaltered. For example, makeoffer(addr2,database) should give as its result:

H(addr1,underoffer) & H(addr2,underoffer)
& H(addr3,forsale) & H(addr4,underoffer) & nil

(4 marks)

(v). Explain the IMAGE feature of the OBJ notation and describe in general terms how it could be employed here in the estate agent system to improve the overall structure of the specification. You are not required to rewrite the specification.

(4 marks)

n

4 (a). Consider the following Z schema specifying the state of a diary system for recording which activity is associated with which time slot. You should regard TIMESLOT and ACTIVITY as given sets that need not be elaborated further.

(i). Give a line-by-line explanation of this schema. (4 marks)

(ii). Provide a Z schema for an operation ShowActivity which extracts from the planned events the activity corresponding to a given time slot, provided that the time slot is valid and one for which an activity is scheduled. Your schema should also report 'Success' as an output.
(3 marks)

(iii). Provide a Z schema FreeSlot which, when given a valid time slot but for which no activity is currently planned, just reports 'Free' as an output.
(2 marks)

(iv). Provide a Z schema InvalidQuery which, when given an invalid time slot, just reports 'Invalid' as an output. (1 mark)

(v). Use the schema calculus to formulate RobustShowActivity, a 'robust' version of the ShowActivity schema. (1 mark)

(vi). Explain the notion of 'variable hiding' and state how this may be used to obtain the precondition of a Z schema. Illustrate your answer by deriving the schema PreShowActivity representing the precondition of the ShowActivity schema - note not the precondition of the RobustShowActivity schema. (4 marks)

Question continues

4.
cont.

(b). (i). Briefly explain the concepts of token marking and transition firing in the Petri net notation. (2 marks)

(ii). If a Petri net with some initial marking is said to be 'live', what does this mean and why is it significant? (2 marks)

(iii). Consider the Petri net given in the diagram. Give a step-by-step explanation of subsequent markings indicating the concepts that are illustrated.
(4 marks)

(iv). If the initial marking had contained a single token at place p2 instead of the single token at place p1, what can be deduced about the Petri net in this case? You should explain your reasoning. (2 marks)

n

5 (a). In the context of program verification, explain the meaning of the following:

(i). partial correctness; (2 marks)

(ii). total correctness. (1 mark)

State, with appropriate explanations, the proof rules for:

(iii). an assignment statement for a simple scalar variable; (2 marks)

(iv). a while loop. (2 marks)

(b). Consider the following VDM specification for raising a given base number to some power n.

```
power( base : Z, n : Z ) p : Z
pre          n  0
post         p = basen
```

Note that Z is the set of integer whole numbers. Consider also the following implementation of power as an Ada function.

```
function power( base, n : integer )
                                    return integer is
        p, i : integer;
begin
        p := 1;  i := 0;
        while i /= n
        loop
                i := i+1;
                p := p*base;
        end loop;
        return p;
end power;
```

(i). Determine a suitable loop invariant for the while loop. (2 marks)

(ii). By inserting appropriate assertions between statements of the Ada implementation, construct a complete proof tableau for the power function. Hence prove

partial correctness with respect to the VDM specification.

(7 marks)

(iii). Determine a suitable loop variant function and hence prove total correctness.

(3 marks)

(iv). Describe in outline how the strategy of replacing a constant by a variable could have been used to derive in a formal manner the Ada implementation for power from its VDM specification. (4 marks)

(v). Despite the fact that the power function is totally correct, under what circumstances might problems ensue when it is run on a real machine.

(2 marks)

n