UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2003

MSc in Computing Science
for Internal Students of the Imperial College of Science, Technology and Medicine

PAPER M1

PROGRAM DESIGN AND LOGIC

Tuesday 29 April 2003, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

Section A *(Use a separate answer book for this Section)*

1a Formalise the following sentences in predicate logic using the predicates

on(X)	to mean X was on the boat
rem(X)	to mean X remained on the boat
ash(X)	to mean X went ashore
hc(X)	to mean X helped with the cooking
hf(X)	to mean X helped with the fishing
h(X,Y)	to mean X helped Y
inj(X)	to mean X was injured
nurse(X)	to mean X was a nurse

i) One of the people on the boat was a nurse and this nurse helped everyone on the boat who was injured.

ii) Everyone who was on the boat either went ashore and helped with the cooking or remained on the boat and helped with the fishing.

iii) None of the people who went ashore were injured, but none of them were nurses and they did not help anyone who remained on the boat.

b

i) Show $Q(a), \neg\exists X\,(Q(X) \wedge N(X)), \neg M(a,b) \rightarrow N(a) \vdash M(a,b)$ using syntactic techniques only. At each stage of the proof clearly state the inference rule and the wffs used.

ii) Show $S1,S2,S3,S4 \vdash \forall X\,(P(X) \rightarrow \exists Z\, M(X,Z))$ using only syntactic techniques and, if required, b(i), above. At each stage of the proof clearly state the inference rule or b(i) and the wffs used. Sentences S1-S4 are given below:

S1	$\forall X\,(P(X) \rightarrow Q(X) \vee S(X))$
S2	$\forall X\,(S(X) \rightarrow \exists Z\, M(X,Z))$
S3	$\neg\exists X\,(Q(X) \wedge N(X))$
S4	$\forall X\,\forall Y\,(\neg M(X,Y) \rightarrow N(X))$

Parts a and b each carry 50% of the marks.

2a Consider the action of a person giving a lift in his car to another person from one location to another. Express the preconditions and postconditions of this action in the situation calculus assuming that the only properties we are interested in are locations of people and cars. Do not forget that in the situation calculus one state changes into another by one action only. You can use the predicate *at(X, L, S)* to denote that X is at location L in state S.

b Consider the logic sentence
$\forall X,L1,L2,S\ (\ at(X, L1, S) \wedge at(X, L2, S) \rightarrow L1=L2)$

i) Give an English reading of this sentence, where the predicate *at* is as explained in part a.

ii) Give an equivalent formalisation of the logic sentence without using the connectives "$\rightarrow$" and "$\vee$".

c Assume that a Prolog database has facts for the relations:

part(P)	P is a part
supplies(M,P):	manufacturer M supplies part P
located_in(M,T)	manufacturer M is located in town T
direct_component_of(P1,P2)	part P1 is a direct component of part P2

There is a Prolog primitive, *findall*, that can be used to construct a list of all solutions to some query:
findall(T,Q,L) will unify L with the list of terms $[T\Phi_1,T\Phi_2,...,T\Phi_K]$ where $\Phi_1,\Phi_2,...,\Phi_K$ are all the different solutions to the Prolog query Q.
As an example of its use, suppose we have facts:

located_in(johnsons,ipswich). located_in(samuels,ipswich).
and these are the only located_in/2 facts mentioning ipswich. The call:
findall(M, located_in(M,ipswich), L)
binds L to [johnsons,samuels].

Using *findall*, and Prolog's negation operator \+, where appropriate, give Prolog definitions of the following relations in terms of the above database relations:

i) component_of(P1,P2)
P1 is a direct component part of P2 or there is a direct component P of P2 of which P1 is a component

ii) supplies_in_same_town(M1,P,M2) manufacturer M1 supplies part P and is located in the same town as manufacturer M2

iii) atomic_part(P) P is a part with no direct component

iv) supplies_all_of(M,T,L) manufacturer M is located in town T and supplies all and only those parts on the list L

Parts a, b ,c carry 35% , 15% , 50% of the marks, respectively.

Section B (*Use a separate answer book for this section*)

1 Consider the following simplified description of production scheduling:

- A product is defined in terms of three tasks, which have to be executed sequentially, so as to manufacture the product.
- A task is defined in terms of a type (a character), and a nominal duration (floating point number). If the task has been scheduled, then it also has a completion time.
- A machine has a type (a character), a speed (a floating point number) and a time until which it will be busy.
- A product may be scheduled on a machine: If there is no outstanding task on that machine, or, if the type of the product's outstanding task and the type of the machine do not match, then an error message is given. Otherwise, the time until which the machine will be busy is increased by the outstanding task's nominal duration multiplied by the machine's speed. The task is scheduled to have finished by that time.
- The outstanding task of a product is the first of its tasks that has not yet been scheduled.
- Completion information about a product can be requested: if its last task has been scheduled, then its completion time is printed, otherwise a message is printed indicating that the task has not yet been completely scheduled.

You may use a predefined class `Time` to represent time:

```
class Time{
public:
    Time add(float);
    static Time now();        // the time now
    . . .
};
```

a Write C++ class declarations (i.e. no function bodies) to support the above. *Hint:* You are *not* expected to use inheritance.

b Write a test function where:

i) `m1` is a machine of type `'a'` and speed `1.1`, while `m2` is a machine of type `'b'` and speed `2.2`, and `m3` is a machine of type `'a'` and speed `3.3`. Also, `p1` is a product requiring a task of type `'a'` and nominal duration `5.5`, followed by a task of type `'b'` and nominal duration `6.5`, followed by a task of type `'a'` and nominal duration `3.5`. Finally, `p2` is a product requiring a task of type `'c'` and nominal duration `7.2`, followed by a task of type `'b'` and nominal duration `6.2`, followed by a task of type `'a'` and nominal duration `3.2`.

ii) Product `p1` is scheduled on machine `m1`, then `p1` is scheduled on `m2`, then `p2` is scheduled on `m3`.

iii) Completion information is requested for `p1`.

iv) Product `p1` is scheduled on `m1`. Completion information is requested for `p1`.

c Write the bodies of the functions from part a.

The three parts carry, respectively, 40%, 10%, 50% of the marks.

2 Consider the following simplified description of telephone billing:

- Telephone calls are characterized by the time the call started and the time the call finished. The duration of a call is the difference between the finish and start time. We distinguish free calls which cost nothing, late calls which cost a quarter of the duration, and normal calls which cost as much as the duration.
- The system tracks the telephone calls of customers. Customers are divided into normal customers and chatterboxes. Chatterboxes are given some free units.
- A telephone call initiated after 22:00 by a normal customer, counts as a late call; otherwise it counts as a normal call. A telephone call initiated after 18:00 by a chatterbox who has more than 0 free units, counts as a late call, and the free units are decremented by 1; otherwise it counts as a normal call.
- When a customer is requested to pay their bill, the details of all their calls are printed (i.e. the start and finish time and the cost), the total of all costs is printed, and their list of calls is set to empty.

You may assume the following predefined classes:

```
template <class T>
class List { // a list of Ts
public:
   List();                 // an empty list
   bool isEmpty();         // whether list is empty
   void insert(T* aT); // inserts  aT  into the list
   T* get();   // returns an element, and removes it from list
};
class Time{
 ...
public:
    bool after18();          // whether it is after 18:00
    bool after22();          // whether it is after 22:00
    double minus(Time&)      // difference between times
}
```

a Develop a UML class diagram to describe the above.

b Write C++ class declarations (i.e. no function bodies) to support the above.

c Write a test function, where, assuming that `t1`, `t2`, `t3`, `t4`, `t5`, `t6` are times:

i) `Jane` is a chatterbox with `300` free units, and `John` is a normal customer.

ii) `John` makes a call from `t1` to `t2`,
`Jane` makes a call from `t3` to `t4`.

iii) `John` is requested to pay, and then `John` makes a call from `t5` to `t6`.

d Write the bodies of the functions from part a.

The four parts carry, respectively, 25%, 25%, 15%, 35% of the marks.