UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1998

MSc Degree in Computing Science
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

PAPER COMP I

PROGRAM DESIGN AND LOGIC
Thursday, May 14th 1998, 10.00 - 12.00

Answer THREE questions
Answer at least ONE question from Section A
Answer at least ONE question from Section B

For admin. only: paper contains 4 questions

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

1 Consider the following simplified version of an adventure game in which lemmings move from room to room:

Each room is connected to another room in the West, and to a further room in the East. The connections are symmetric, i.e. if room A is connected to room B in the West, then room B is connected to room A in the East.

Lemmings share the same DLM, Direction of Lemming Movement, which may be East or West. The DLM is initially West, but it may be changed.

When a lemming is woken up, then it moves to the room which is in the direction of the DLM, from the room it is currently in.

a Draw an OMT object model class diagram describing the above. (A summary of the OMT object model notation can be found at the end of this section).

b Write C++ classes that support the above.

c Write a main program that

i) creates three rooms r1, r2, r3,

ii) connects r2 to the West of r1, r3 to the West of r2, and r1 to the West of r3,

iii) creates lemming Lala in room r1, and lemming Lilo in room r3,

iv) wakes up Lilo, then wakes up Lilo again, wakes up Lala,

v) sets the DLM to West.

vi) wakes up Lilo.

The three parts carry, respectively, 20% , 60% and 20% of the marks.

2 Consider interest paying accounts, where interest is added to the balance on each transaction, and it is calculated according to the following formula:

Balance on the day of latest transaction * 0.0003 * Interest Rate * (Number of days between Current Transaction and Previous Transaction)

We distinguish `Golden Accounts` and `Privileged Accounts`: The interest rate for all `Golden Accounts` is set centrally, whereas for `Privileged Accounts` the interest rate is agreed upon on account creation.

a Write C++ classes (interface and implementations) to describe both kinds of accounts.

For `Golden Accounts` provide the possibility to:

i) set the interest rate for all accounts to a new value;

ii) create a new account on a certain date with an initial balance;

iii) deposit an amount of money on a certain date: (calculate and add interest, and then increase the balance);

iv) withdraw an amount of money on a certain date: (calculate interest, decrease the balance);

v) enquire the balance of the account.

For `Privileged Accounts` provide the possibility to:

i) create a new account on a certain date, with an interest rate and with an initial balance;

ii) deposit an amount of money on a certain date: calculate interest, and then increase the balance;

iii) withdraw an amount of money: calculate interest, and then decrease the balance;

iv) enquire the balance of the account.

b Write a main program in which

i) the interest rate for `Golden Accounts` is set to 7.0;

ii) the variable `GA` contains a `Golden Account` created on the 10th February 1993, with 140.00 initial balance;

iii) the variable `PA1` contains a `Privileged Account` created on the 25th February 1993, with 100.00 initial balance, and interest rate of 6.5;

iv) the variable `PA2` contains a `Privileged Account` created on the 17th March 1993, with 80.00 initial balance, and interest rate of 7.5;

v) the amount of 60.00 is paid into `GA` on the 16th June 1994;

vi) the interest rate for `Golden Accounts` is set to 6.0.

The two parts carry, respectively, 80% and 20% of the marks.

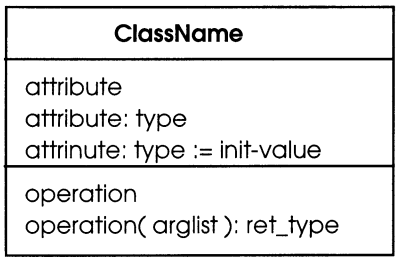
Note: For the description of dates use the following definitions:

```
enum month {   jan, feb, mar, apr, may, jun,
               jul, aug, sep, oct, nov, dec};
class Date { public:
    Date(int d=1; month m=jan; int y=1996);
    int operator -(Date);
    // returns number of days between receiver and argument
..};
```

Turn over ...

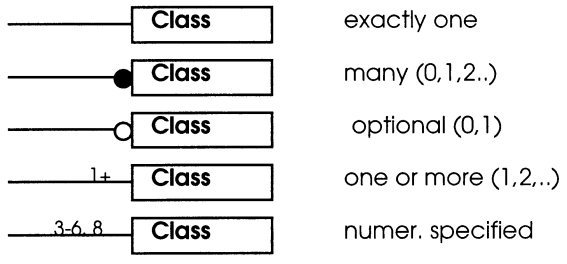
OMT: Basic Notation for Object Models

Class:

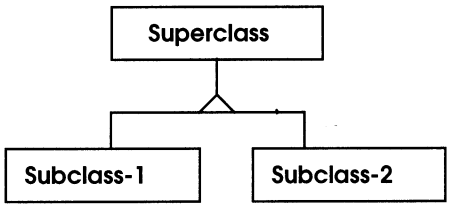


Association

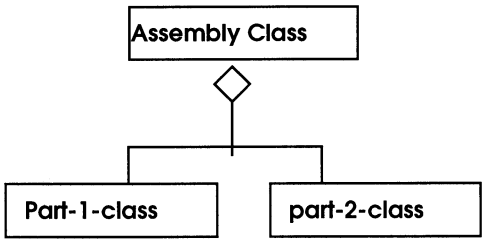
Multiplicity of Association/Aggregation



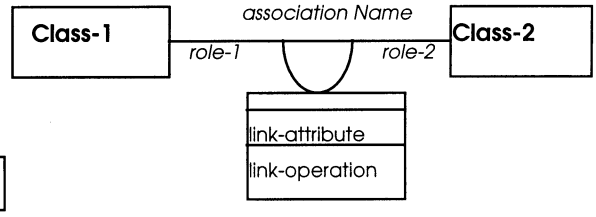
Generalization (Inheritance)



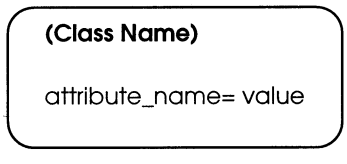
Aggregation



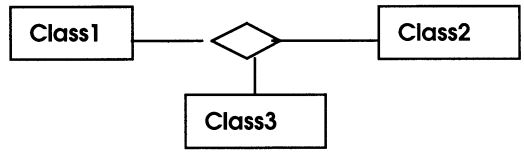
Link Attributes



Object Instance



Ternary Association



$ for class operations/attributes

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

3a Formalise the following sentences in predicate logic, using the following predicates

person(X) :	X is a person
inroom(X) :	X is in the room
knows(X, Y) :	X knows Y
film(X) :	X is a film
likes(X, Y) :	X likes Y
infilm(X, Y) :	X is in film Y
seen(X, Y) :	X has seen Y

i) No one in the room except Mary knows John, but everyone there knows Harry.

ii) None of the people who knows Harry knows anyone who knows either Mary or John.

iii) John has seen all the films that Harry has been in, but John likes only two of them.

iv) Not everyone in the room likes all the films that Harry has been in, and no one there likes any of the films that Mary likes.

b During the party Mary tried to introduce John, Ken and herself to someone. She started by saying:

> "One and only one of us (Mary, John, Ken) is an actor, and one and only one of us is a dentist, but none of us is both an actor and a dentist."

The rest of what she said can be formalised in propositional logic as follows:

$MA \rightarrow \neg JD$

$KA \rightarrow KD$

$\neg KA \wedge \neg KD \rightarrow \neg MD$

$\neg JD \wedge MA \rightarrow \neg KD$

where MA (or JA, or KA) means Mary (or John, or Ken) is an actor, and MD (or JD, or KD) means Mary (or John, or Ken) is a dentist.

Using the above formalisation, the inference rules of natural deduction and formalising as much of Mary's starting statement as you need, show which of the three individuals is an actor and which is a dentist. Do not use equivalences in your proof. Explain every step of your proof.

Turn over ...

4a i) Using only natural deduction inference rules (and no equivalences) show

$\forall X\ (P(X) \rightarrow Q(X))\ \vdash\ \exists X\ P(X) \rightarrow \exists X\ Q(X)$.

Explain every step of your proof.

ii) Using any syntactic or semantic techniques show

$\forall X\ (A(X) \rightarrow \exists Y\ B(X,Y)) \equiv \neg\exists X\ (A(X) \wedge \forall Y\ \neg B(X,Y))$.

iii) Using only natural deduction inference rules and, if required, parts (i) and (ii) above, show

$\neg\exists X\ (P(X) \wedge \forall Y\ \neg Q(X,Y)), \forall X\forall Y\ (Q(X,Y) \rightarrow R(Y))\ \vdash$

$P(a) \wedge \neg R(a) \rightarrow \exists Y\ \neg Y{=}a$.

Explain every step of your proof.

b i) Write a Prolog program for relation

count(Term, List, Number)

such that *Number* is the number of occurrences of the term *Term* in the list *List*. You may assume that in any query concerning *count* the first argument will be a constant and the second argument will be a list of constants.

ii) Write a Prolog program for relation

process(List1, List2)

such that *List2* is a list consisting of items of the form *[Term, Number]*, one such item for each term *Term* that occurs in *List1*, and *Number* is the number of occurrences of that term in *List1*. *List 2* should contain no repetitions. For example the query *process([a,b,a,c,c], L)* should give the answer *L= [[a,2],[b,1],[c,2]]*. You may assume that in any query concerning *process* the first argument will be a list of constants.

End of paper