UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2001

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

PAPER M1

PROGRAM DESIGN AND LOGIC

Tuesday 15 May 2001, 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 Translate the following sentences (i)-(iv) into predicate logic, using the following predicates:

gives(X,Y,Z): X gives object Y to person Z as a gift

friend(X,Y): X is a friend of Y

edible(X): X is edible

drinkable(X): X is drinkable

bof(X): X is a basket of fruits

i) Any gift Janet gives to any of her friends is either edible or drinkable, but it is never a basket of fruits.

ii) Peter gives gifts only to those people who do not receive gifts from anyone else.

iii) If Peter gives an edible gift to someone he will not give them any other gifts, unless the edible gift is a basket of fruits.

iv) Janet's friends all give gifts to one another and to Janet, but they never give anyone a basket of fruits as a gift.

b Consider the following sentence of logic:

$\neg\exists X,Y$ [gives(janet,X,jim) $\wedge$ gives(peter,Y,jim) $\wedge$ $\neg$drinkable(X)].

i) Reformulate the sentence to an equivalent one which has no existential ($\exists$) quantifiers.

ii) Give an English reading of the sentence. Make the English as natural as possible.

c Using the situation calculus, formalise preconditions and postconditions for the act of "giving something to someone". Assume that the only property that is to be taken into account is the possession of objects (i.e. who has what object).

The three parts carry, respectively, 50%, 25%, 25% of the marks.

2a i) The following rule of inference is called Dilemma:

$$\frac{X \rightarrow Y,\ \neg X \rightarrow Y}{Y}$$

Prove this rule using any other rules of inference except Proof By Cases. Do not use any equivalences. At each stage of the proof clearly state the inference rules and the wffs used.

ii) Prove the following equivalence using any semantic or syntactic techniques that you wish:
$\neg C \rightarrow (A \wedge \neg D \rightarrow B) \equiv A \wedge \neg B \rightarrow C \vee D$

iii) Given the following statements S1-S6:

S1	$\neg C \rightarrow (A \wedge \neg D \rightarrow B)$
S2	$B \rightarrow E$
S3	$\neg E$
S4	$C \wedge F \rightarrow G$
S5	$C \wedge \neg F \rightarrow H$
S6	$D \rightarrow G$
prove	$A \rightarrow G \vee H$

using any inference rules, and, if required, the equivalence in (ii). Do not use any other equivalences. At each stage of the proof clearly state the inference rules and the wffs used.

b A *tree* is an atom, or of the form tree(T1,T2) where T1 and T2 are trees. Thus, tree(tree(a,b),c) is a tree. The leaf profile of such a tree is the list of atoms appearing at its tips, in left to right order. Thus, [a,b,c] is the leaf profile of tree(tree(a,b),c) and of tree(a,tree(b,c)). The leaf profile of a tree that is an atom A, is the list [A].

Consider the relation:

leaf_profile(T,L) L is the list of atoms which is the leaf profile of tree T.

i) Give a two-clause recursive definition of this relation that uses the app relation defined by:

```
app([],Y,Y).
app([U|X],Y,[U|Z]) :- app(X,Y,Z).
```

You can also use the Prolog primitive, atom(A), for testing if a term is an atom.

ii) Give a three-clause recursive definition of the relation that does not use the app relation. Your definition should only use recursive calls to leaf_profile and the atom test primitive.

[Hint: treat separately the cases: left subtree is an atom, left subtree is a tree.]

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

3 Consider the following simplified description of portfolios:

A *portfolio* holds share accounts, and charges fees upon some transactions. Fees may change. *Share accounts* track a (floating point) number of shares of a company. *Companies* have a share price, which may change.

When an amount is paid into a share account, the amount is reduced by the portfolio's fee, and then the number of shares is increased by the (reduced) amount divided by the company's share price. When an amount is withdrawn from a share account, first the amount is increased by the portfolio's fee, and then the number of shares is decreased by the (increased) amount divided by the company's share price.

The balance of a share account is the number of shares multiplied by the company's share price. The balance of a portfolio is the sum of the balances of its share accounts.

a Develop an OMT class diagram to describe the above. (You can find a summary of the OMT notation on the last page of this paper.)

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

c Write a test function that:

i) creates a portfolio P1 with a fee of 3.3, and a portfolio P2 with a fee of 2.2;
ii) creates company BT with a share price of 60.0;
iii) creates SA1, a share account held by P1, tracking shares of BT;
iv) creates SA2, a share account held by P2, tracking shares of BT;
v) pays 444.44 into SA1, and pays 555.5 into SA2;
vi) sets the share price of BT to 3.0;
vii) enquires about the balance of P1.

d Write the body of the function that calculates the balance of a portfolio.

Note: For parts b and d you may assume the existence of the following template class:

```
template <class T>
class List { // a list of T*s
public:
   List();                  // an empty list of T*s
   void insert(T* aT);      // inserts aT into list
   T* first();              // first element in list
   T* next(T* aT);          // element in list after aT
...};
```

The four parts carry, respectively, 35%, 35%, 10%, 20% of the marks.

4 Consider a simplified description of books and periodicals held in a library:

Books and periodicals have a title and a location. The locations are numbers, allocated consecutively upon arrival at the library, *e.g.* the first book or periodical has location 1, the next book or periodical has location 2 etc.

Books and periodicals may be requested for loan for a specified number of days.

When a book is requested for loan, then, if the book is not already on loan, and the number of days requested is smaller than the maximum loan period for books, then the book is marked as being on loan and its location is returned. Otherwise, an error message is printed.

The maximum loan period is the same for all books and may be changed from time to time by the library administration.

When a periodical is requested for loan, then, if the periodical is not already on loan, and if the number of days requested is smaller than the maximum loan period for this periodical, then the periodical is marked as being on loan and its location is returned. Otherwise, an error message is printed.

The maximum loan period for periodicals is 5 days by default, but it may be explicitly changed for particular periodicals at any time.

a Create an OMT class diagram describing the above. (You can find a summary of OMT notation on the last page of this paper.)

b Write C++ class headers which support the above.

c In the main program,

i) set the maximum loan period for books to 10 days;

ii) define a book B1 with the title "Prolog is great"; define a periodical P1 with the title "Haskell is better" and maximum loan period 4; define book B2 with the title "C++ is the best";

iii) set the maximum loan period for P1 to 44 days;

iv) ask to borrow B1 for 6 days; ask to borrow B2 for 11 days;

v) set the maximum loan period for books to 15 days;

vi) ask to borrow B2 for 11 days.

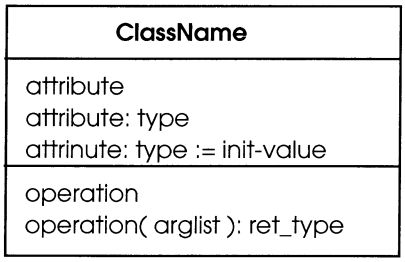
d Write C++ function bodies for the functions introduced in part b.

Note: Part b and part d are marked separately, but, if you prefer, you can write the function bodies inline.

The four parts carry, respectively, 30%, 30%, 15%, 25% of the marks.

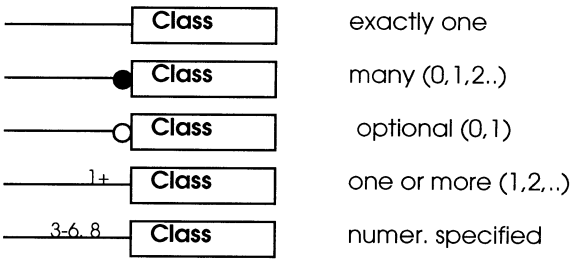
OMT: Basic Notation for Object Models

Class:

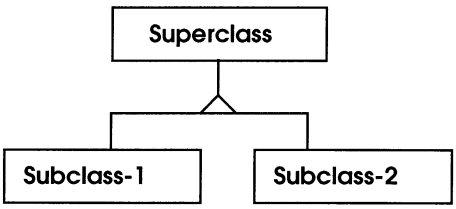


Association

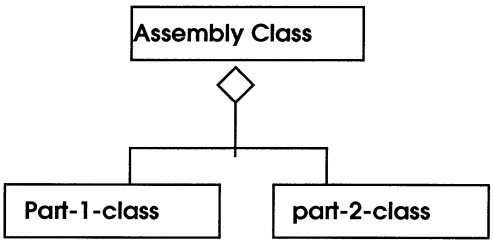
Multiplicity of Association/Aggregation



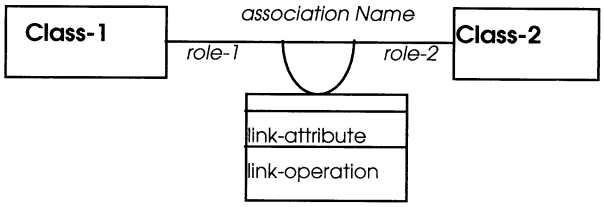
Generalization (Inheritance)



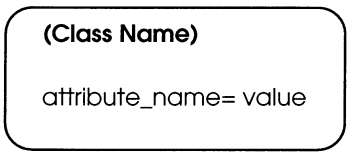
Aggregation



Link Attributes



Object Instance



$ for class operations/attributes

Ternary Association