UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1997

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

DESIGN, LOGIC AND DECLARATIVE PROGRAMMING
Friday, April 25th 1997, 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 5 questions

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

1 Consider expressions, like `3*(4+2)+aNumber`, where an expression is either an integer number, or an identifier, or the sum of two subexpressions, or the product of two subexpressions.

The value of a sum is calculated by adding the values of the two subexpressions. The value of a product is calculated by multiplying the values of the two subexpressions. The value of a number is the number itself, whereas the value of an identifier is found by looking it up in a table.

a Draw an OMT object model class diagram describing expressions. (On the next page you will find a summary of the OMT object model notation).

b Write C++ classes that support the creation of expressions and the calculation of their values.

Use the following class:

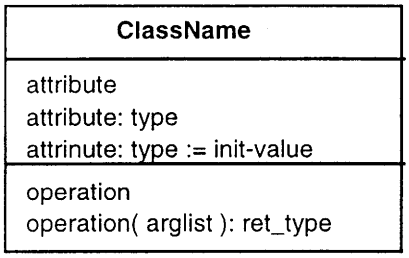
```
class table{
public:
        int lookUp(char* ident);
        table();
};
```

c Write a main program that creates the expression `3*(aNum1+aNum2)` and evaluates it.

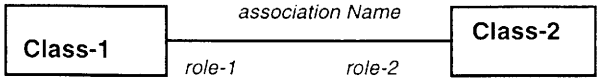
The three parts carry, respectively, 20% , 60% and 20% of the marks.

OMT: Basic Notation for Object Models

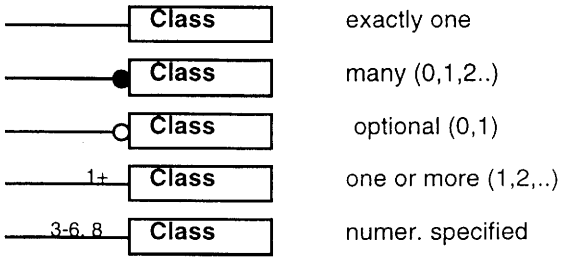
Class:



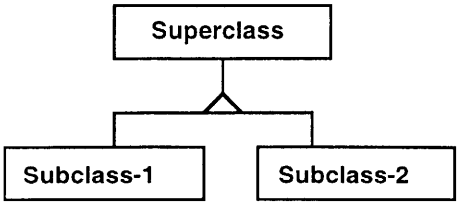
Association



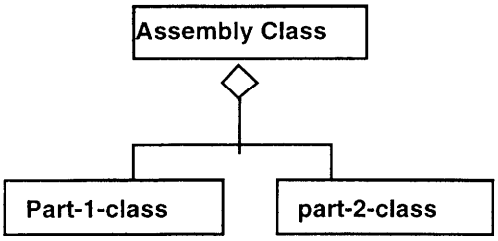
Multiplicity of Association/Aggregation



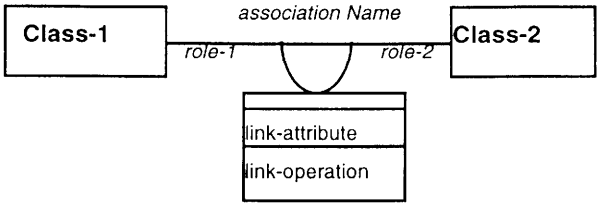
Generalization (Inheritance)



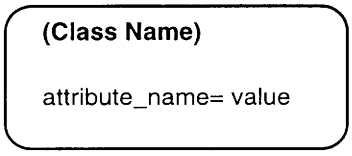
Aggregation



Link Attributes

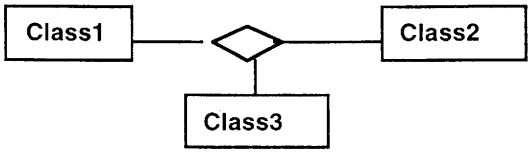


Object Instance



$ for class operations/attributes

Ternary Association



Turn over ...

2a A company which produces a large number of utility programs in C++ is trying to standardise the way in which help texts are provided. To this end it wishes to define a class definition which is to be incorporated into every program. This should provide a unique name for the program, a version number (an integer which need not be further defined), the name of the author(s) and a one-line definition of the function of the program. Facilities should be provided to access each of these fields from outside the class in a safe manner and to print the definition.

Write a class declaration named "ProgramHelp" in C++ for this class. You do *not* need to provide a definition of any of the methods.

b. A program is to be provided which takes a string and searches a database of such definitions to print out the name and one-line description of any definition which contains the string. Write a function "lookup" which takes as parameters the string to be looked for, an array of objects of type ProgramHelp and the length of the array. You may call the methods of the class but you should *not* call any other built-in C or C++ functions.

c Explain how objects are protected from outside interference in C++ and explain briefly what this contributes to the development of reliable software.

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

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

4a Formalise the following sentences in predicate logic, using the following predicates

ac(X) :	X is a member of the academic staff
l(X), ra(X), ta(X) :	X is a lecturer, an RA, a TA, respectively
t(X, Y) :	X teaches course Y
phD(X) :	X is doing a PhD
h(X, Y) :	X helps in the tutorial of course Y
w(X, Y) :	X works on research project Y
s(X, Y) :	X is supported by grant Y
holds(X, Y) :	X holds grant Y

i) A member of the academic staff is either a lecturer, a research assistant (RA) or a teaching assistant (TA).

ii) Each lecturer teaches at least one course.

iii) TAs help in the tutorials of courses, but no TA who is also doing a PhD helps in the tutorial of more than one course.

iv) All RAs work on a research project supported by a grant held by a lecturer.

v) Any member of the academic staff either teaches at least one course or works on a research project, if he/she does not help in the tutorials of any course.

b i) Show $X \rightarrow Z,\ Y \rightarrow Z,\ X \vee Y \ \vdash\ Z$, using any inference rules of propositional logic except "proof by cases".

ii) Show $\neg X,\ X \vee Y \ \vdash\ Y$, using any inference rules of propositional logic except $\vee$-elimination. You may also use b(i).

iii) Given

P1: $\forall X\ [\neg P(X) \rightarrow (Q(X) \vee \exists Y\ R(X, Y))]$

P2: $\forall X[\ P(X) \rightarrow S(X)]$

P3: $\neg S(a)$

P4: $\exists X\ Q(X) \rightarrow \forall X\ (P(X) \vee S(X))$

P5: $\forall X\ \forall Y\ [\ R(X, Y) \rightarrow T(X)]$,

where "a" is a constant symbol, show

P1, P2, P3, P4, P5 $\vdash\ \exists X\ T(X)$, using any inference rules of predicate logic.

5a i) Show the following equivalence using any combination of syntactic and semantic techniques, except truth tables.

$(P \rightarrow (Q \wedge R)) \rightarrow R \equiv (\neg P \vee Q) \rightarrow (P \vee R)$.

ii) Show

$(\neg P \vee Q) \rightarrow (P \vee R),\ \neg R \vdash P,$

using only the inference rules of propositional logic, and, if necessary, the equivalence in (i).

b In this part of the question you are allowed to use the Prolog built-in predicate *append* and the Prolog connective *not*.

i) Write a Prolog program for relation

adjacent(Y, Z, X)

that succeeds when and only when Y and Z are adjacent elements of list X, with Y preceding Z.

ii) Write a Prolog program for relation

delete(E1, E2, X, Y)

such that Y is the result of deleting adjacent elements E1 and E2 from list X. Your program should fail if E1 and E2 are not adjacent elements of X.

iii) Write Prolog programs for relations

even(X) and odd(X)

such that the former succeeds if and only if X is a list with an even number of elements and the latter succeeds if X is a list with an odd number of elements. Your program should *not* compute the length of any list.

End of paper