UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2000

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

PAPER M1

PROGRAM DESIGN AND LOGIC

Friday 12 May 2000, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions

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

1 Consider the following aspects of the "Containers Game", where bullets are thrown into containers, containers are connected to further containers, and excessive bullets overspill to connected containers. The game terminates as soon as a bullet reaches the final container:

- Bullets have a weight, which does not change.
- Containers may be weight containers, capacity containers, or final containers.
- A weight container has a maximal weight, and is connected to a further container. When a bullet reaches a weight container, if the total weight of the bullets exceeds the maximal weight of that container, then the bullet that entered the container the least recently overspills to the connected container.
- A capacity container has a maximal capacity, and is connected to a further container. When a bullet reaches a capacity container, if the total number of the bullets exceeds the maximal capacity of that container, then the bullet that entered the container the least recently overspills to the connected container.
- When a bullet reaches a final container, the game terminates.

a Draw an OMT object model class diagram describing the above. (You will find a summary of the OMT notation on page 3 of this exam paper).

b Write C++ classes (i.e. declarations and function bodies) to implement the above. You may use `FiFoQueue<T>` a template class for first-in-first-out-queues:

```
template <class T>
class FiFoQueue { // a first-in, first-out queue of Ts
public:
    FiFoQueue();
    // an empty first-in, first-out queue of Ts
    void insert(T& aT);
    // inserts aT into the queue
    T& get();
    // removes that entered the queue the least, and returns it
    // throws an exception if queue is empty
...};
```

c Write a test function where:

i) F1 is a final container; C1 is a capacity container connected to F1 and has a maximal capacity of 12; W1 is a weight container connected to F1 and has a maximal weight of 3.3; W2 is a weight container connected to W1 and has a maximal weight of 4.4;

ii) B1 is a bullet with weight 1.1, B2 is a bullet with weight 1.6.

iii) Throw B1 into W1; throw B2 into W2.

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

2 Consider the following simplified description of queues of processes, where:

Every process has a priority and a workload.

Queues provide the capability to:

- create a new queue,
- insert a process into a queue,
- remove and return the process most recently entered into the queue.

Furthermore, we distinguish priority queues from workload queues:

- A process may be inserted into a priority queue only if its priority is higher than that of the process most recently entered in the queue; otherwise an error message is printed.
- A process may be inserted into a workload queue only if its workload is higher than that of the process most recently entered in the queue; otherwise an error message is printed.

Note that a process may simultaneously belong to several different queues; removing a process from one queue does not affect any other queue to which it may belong.

a Develop an OMT object model class diagram to describe the above. (You will find a summary of the OMT notation on the 3rd page of this exam paper).

b Write C++ classes (i.e. declarations and function bodies) to support the above.

c Write a test function, where

i) P1 is a process with priority 3 and workload 4.4, P2 is a process with priority 5 and workload 8.8, P3 is a process with priority 7 and workload 2.5.

ii) PQ is a priority queue; insert P1 into PQ; insert P2 into PQ, then insert P3 into PQ.

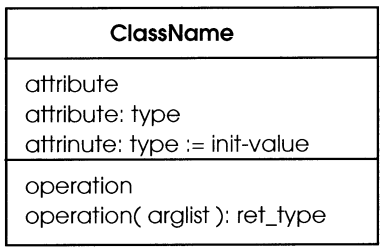
iii) WQ is a workload queue; insert P1 into WQ; then insert P3 into WQ.

iv) Remove from WQ the process most recently entered, and assign it to P4.

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

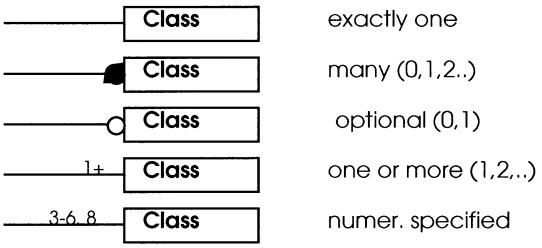
OMT: Basic Notation for Object Models

Class:

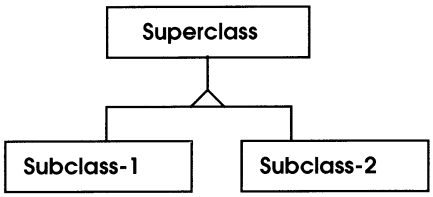


Association

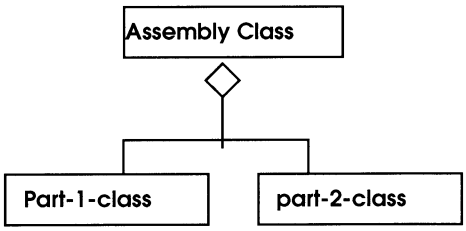
Multiplicity of Association/Aggregation



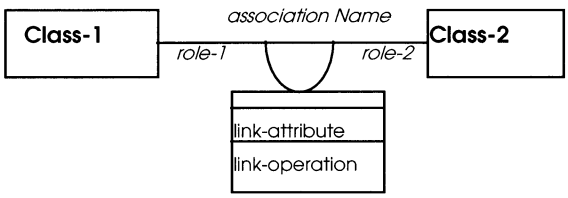
Generalization (Inheritance)



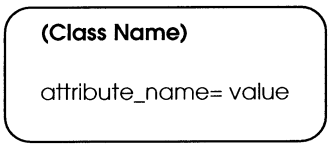
Aggregation



Link Attributes

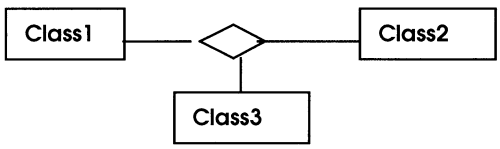


Object Instance



$ for class operations/attributes

Ternary Association



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

3a Translate the following sentences (i)-(iv) into predicate logic, using the following predicates. Sentences (i)-(iv) describe fictional rules regarding welfare provisions.

person(X):	X is a person
age(X,Y):	X is Y years old
X=<Y:	X is less than or equal to Y
	(similarly X>=Y, X<Y, X>Y)
woman(X):	X is a woman
employed(X):	X is employed
child(X,Y):	X is a child of Y
entitled(X,Y):	X is entitled to Y
in_ed(X):	X is in full-time education

i) All unemployed people are entitled to *Unemployment Benefit*.

ii) Any unemployed woman who has a child under the age of 16 is entitled to *Child Allowance*.

iii) Everyone between the ages of 16 and 21, inclusively, is entitled to a *Young Person's Allowance* provided they are neither employed, nor in full-time education.

iv) A woman who is employed is not entitled to *Child Allowance* if she has no child under 14, and any children of hers who are between 14 and 18, inclusively, are either employed or are in full-time education.

b i) Show the following equivalence using any syntactic or semantic techniques.

$$\forall X\,[p(X) \rightarrow \exists Y q(X,Y)] \equiv \neg\exists X\,[p(X) \wedge \forall Y\, \neg q(X,Y)]$$

ii) Using only inference rules, and, if required, the equivalence above, show
P1, P2, P3 $\vdash \neg\exists X\,[p(X) \wedge \forall Y\, \neg q(X,Y)]$
where P1-P3 are as follows. Explain every step of your proof clearly by giving the inference rules and the wffs used.

P1 $\forall X\,(p(X) \rightarrow t(X) \wedge s(X))$
P2 $\forall X\,\forall Y\,(t(X) \wedge m(X,Y) \rightarrow q(X,Y))$
P3 $\forall X\,((s(X) \rightarrow \exists Y m(X,Y))$

Parts a and b carry 60% and 40% of the marks, respectively.

4 a i) Using any inference rules, except Dilemma and Proof by Cases, show $X \rightarrow Y, \neg X \rightarrow Y \vdash Y$.
Explain every step of your proof clearly by giving the inference rules and the wffs used.

ii) Using inference rules only, show
$X \rightarrow Y \vdash \neg X \vee Y$.
Explain every step of your proof clearly by giving the inference rules and the wffs used.

iii) Using only inference rules, and, if required, 4a(i), 4a(ii), above, and the equivalence $\neg\neg X \equiv X$, show

$P1, P2, P3, P4, P5 \vdash C \vee D$

where P1, P2, P3, P4, P5 are as follows:

P1 $A \rightarrow (B \rightarrow (\neg C \rightarrow D))$
P2 $A \rightarrow (\neg B \rightarrow (\neg C \rightarrow D))$
P3 $\neg A \rightarrow E$
P4 $\neg (E \wedge F)$
P5 F

Explain every step of your proof clearly by giving the inference rules and the wffs used.

b i) Write a Prolog program for the predicate *drop(L, N, NewL)*, such that, *NewL* is list *L* with its first *N* elements deleted. So, for example the query *drop([1,4,7,1], 2, NewL)* produces the answer *NewL=[7,1]*. You may use Prolog's built-in predicates >=, *append*, *length*, but no others.

ii) Modify your program for *drop* to write a program for *drop_more(L, N, NewL)*, such that, *NewL* is list *L* with its first *N* elements, and all their duplicates deleted. So, for example the query *drop_more([1,4,7,1], 2, NewL)* produces the answer *NewL=[7]*. You may use Prolog's built-in predicates >=, *append*, *length*, *delete*, but no others. Assume that the Prolog built-in predicate *delete(L, E, NL)* deletes all occurrences of single element *E* from list *L* to produce list *NL*.

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

End of paper