UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2006

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
MSc in Computing Science
MSc in Computing for Industry
BEng Honours Degree in Information Systems Engineering Part III
MEng Honours Degree in Information Systems Engineering Part III
BSc Honours Degree in Mathematics and Computer Science Part II
MSci Honours Degree in Mathematics and Computer Science Part II
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute
This paper is also taken for the relevant examinations for the Associateship of the Royal College of Science

PAPER C223=MC223=I3.27

CONCURRENT PROGRAMMING USING JAVA

Tuesday 25 April 2006, 15:00
Duration: 120 minutes

Answer THREE questions

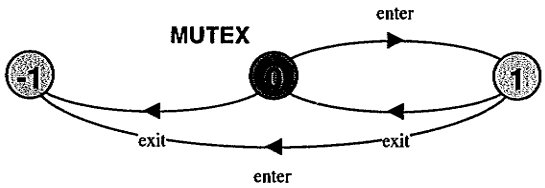
Paper contains 4 questions
Calculators not required

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

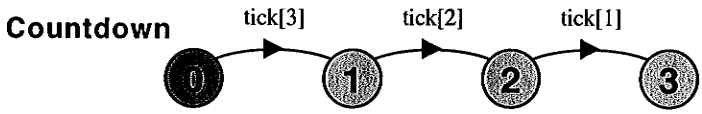
1a Processes modeled in FSP are said to execute *asynchronously*. Briefly explain what *asynchronous* execution means in this context.

b For each of the following Labelled Transition Systems (*LTS*), give an equivalent *FSP* specification.

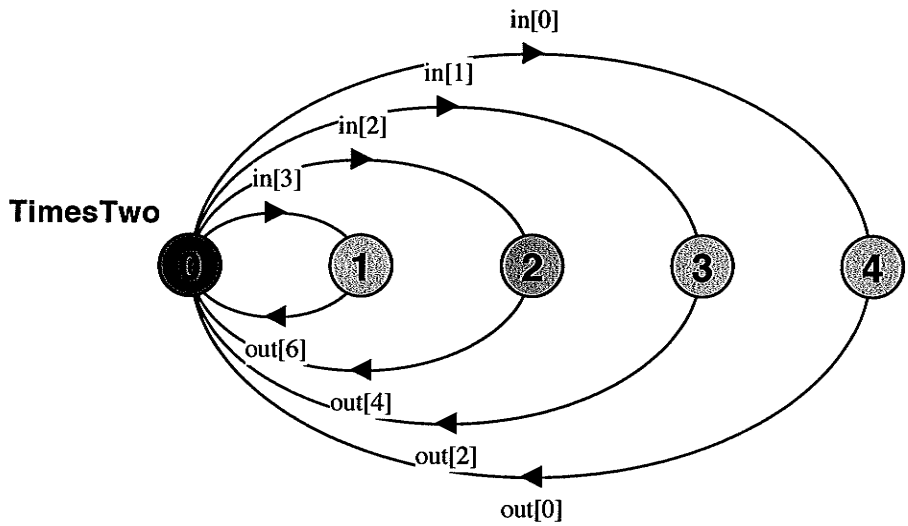
i)



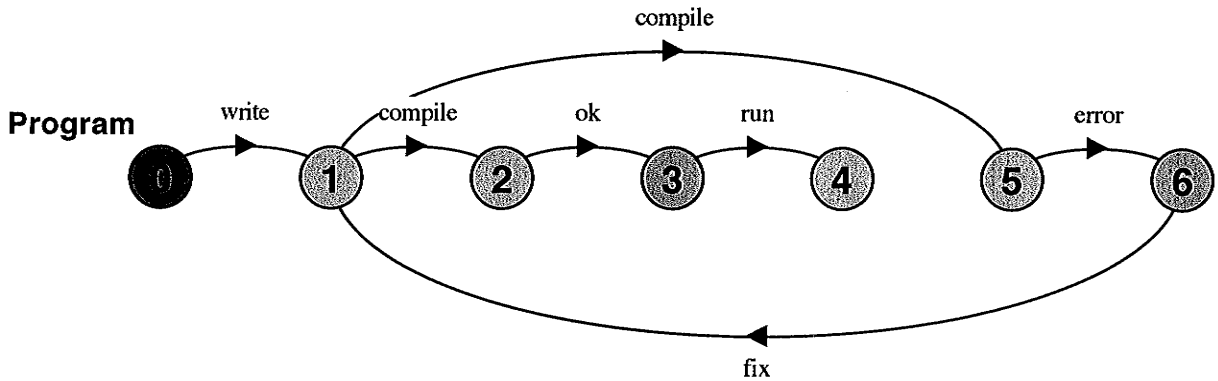
ii)



iii)



iv)



c For each of the following *FSP* specifications, give an equivalent *LTS*.

i)
```
set A = {heads, tails}

HeadsOrTails = PICK,
PICK = (pick[i:A] -> FLIP[i]),
FLIP[i:A] = ( flip[j:A] -> RESULT[i][j]),
RESULT[i:A][j:A] = (when (i==j) win -> HeadsOrTails|
                    when (j!=i) lose -> HeadsOrTails).
```

ii)
```
property SAFE = (open->OPEN),
OPEN = ( enter->OPEN | close->SAFE).
```

iii)
```
CONSTRAIN = STOP + {north}.
TURTLE  = (north->move->TURTLE | south->move->TURTLE).
||SYS = (CONSTRAIN || TURTLE).   //draw LTS for SYS
```

iv)
```
GATE  = (in->out->GATE).
||AND = (a:GATE || b:GATE)/{out/{a.out,b.out}}.
//draw LTS for AND
```

The three parts carry, respectively, 20%, 40%, 40% of the marks.

2a

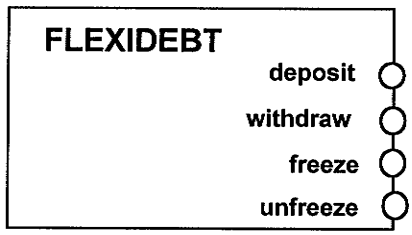
i Explain what the keyword **`synchronized`** and the method **`wait()`** do when used in Java for condition synchronisation in monitors.

ii In the following code, if a thread were blocked at ***wait statement***, and another thread attempts to execute **`foo()`**, would the hello world message be printed? Explain.

```
synchronized foo() {
        system.out.println("Hello World!");
        ...
        wait(); // wait statement
        ...
}
```

b A Flexible Loan Account is permitted to have a maximum debt of **M** hundred Pounds. It is possible to withdraw money one hundred pounds at a time as long as the debt does not exceed the limit. It is also possible to pay back debt in hundred pound installments. However, the user of an account can be prevented from acquiring more debt if the bank administrators consider there are additional risks of the user defaulting on the debt. When this is the case, the account is said to be frozen.

The alphabet of the process that models the savings account is depicted below, together with a definition of the meaning of each action.



range B= 0..M

deposit	- repay one hundred pounds of debt. This action is blocked if the balance would exceed **0**.
withdraw	- withdraw one hundred pounds. This action is blocked if the debt were to exceed maximum allowed debt or if the account were frozen.
freeze	- Freezes the account, preventing new debt to be taken.
unfreeze	- Unfreezes the account, allowing new debt to be taken.

Specify the behaviour of FLEXIDEBT in FSP.

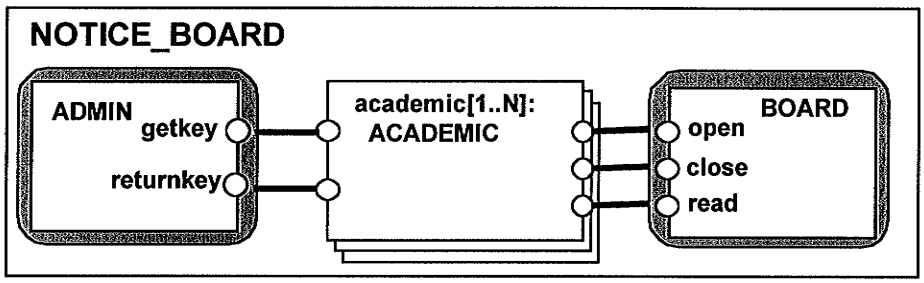
c Implement the FLEXIDEBT specification from part b with the four actions as monitor methods programmed in Java.

The three parts carry, respectively, 20%, 40%, 40% of the marks.

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

3a Explain briefly how a resource, shared by a set of processes can be modelled in FSP.

b A notice board is shared by a department of academics. The notice board has a glass cover that may only be opened with the aid of a key obtained from the administrator. To change a notice, an academic must obtain the key, open the glass cover, change the notice, close the cover and return the key. Other academics can read the notice board when it is closed (i.e. not opened for a change). The structure diagram for an FSP model of the system with *N* academics is shown below:



Given that the behaviour of **ACADEMIC** is defined by:

```
ACADEMIC = (getkey ->open ->close ->returnkey ->ACADEMIC
           |read   ->ACADEMIC).
```

specify the behaviour of each of the processes (**ADMIN**, **BOARD**) and the composite process **NOTICE_BOARD** in FSP.

c Implement the specifications for each of the entities (**ADMIN**, **ACADEMIC**, **BOARD**) in Java. Include the definition of a method **void build(int N)** which creates the objects required for **NOTICE_BOARD**.

(Hint: Use the method **Math.random()** which randomly returns a floating point value between zero and one to decide whether an academic chooses to change a notice or simply read the notice board. Ignore details of data representations for keys, notices etc. and use **void** methods with no parameters to implement model actions.)

The three parts carry, respectively, 20%, 30%, 50% of the marks.

4a Explain what is meant by liveness and progress properties with respect to concurrent programs. Give an example, using natural language, of each sort of property.

b A remote login system should 1) allow up to 4 remote users logged in at the same time and 2) ensure that only enabled users are allowed to log in.

The action `login[u]` models the successful login by user `u`, while `logout[u]` indicates that user `u` has logged out where `u` is in the range `U=1..N`. Similarly actions `enable[u]` and `disable[u]` are used by administrator to configure users that are allowed to login to the system.

Specify the two requirements set out above using safety properties in FSP. (*Hint: One of these properties is best specified using a parallel composition of simpler properties*)

c i) Explain what is meant by recursive locking in Java.

ii) Given the following declarations:

```
const N = 3
range P = 1..2  //thread identities
range C = 0..N  //counter range for lock
```

Model a Java recursive lock as the *FSP* process RECURSIVE_LOCK with the alphabet `{acquire[p:P],release[p:P]}`.
`acquire[p]` acquires the lock for thread `p` and `release[p]` releases it.

The three parts carry, respectively, 20%, 40%, 40% of the marks