UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2005

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
MSc in Computing for Industry
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

# CONCURRENT PROGRAMMING

Thursday 12 May 2005, 10:00
Duration: 120 minutes

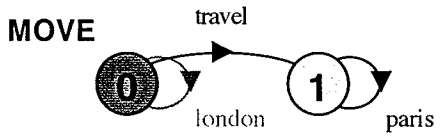
*Answer THREE questions*

Paper contains 4 questions
Calculators not required

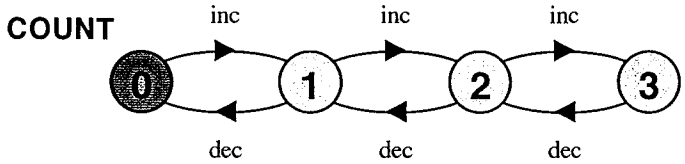
1a Explain how *non-deterministic choice* is expressed in the Finite State Processes (FSP) modelling notation. Briefly explain why it is useful in modelling systems, giving an example to illustrate your answer.

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

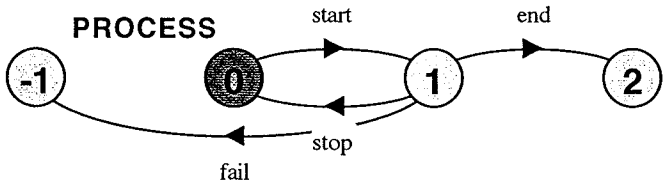
i)



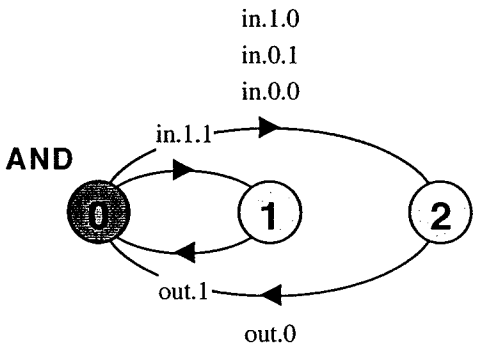

ii)




iii)




iv)




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

i)
```
GAME = (choose[i:1..3] -> (when (i==2) win -> STOP)).
```

ii)
```
CLOCK(N=4) = S[0],
S[i:0..3]  = (tick[i] -> S[(i+1)%N]).
```
*// % is modulus*

iii)
```
DOG     = (bark -> attack -> DOG).
||DOGS = (fido:DOG || rover:DOG)
          /{bark/{fido,rover}.bark}.
```
*// draw LTS for DOGS*

iv)
```
BROKEN  = STOP + {tick[2]}.
||ENDOFTIME = (CLOCK(20) || BROKEN).
```
*//draw LTS for ENDOFTIME, CLOCK is as defined in ii) above.*

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

2a Briefly explain why *alphabet extension* is useful for modelling concurrent programs. Provide and example of its use.

b The interface to a buffer that stores characters is specified in Java as follows:

```
interface Buffer {

  public static int N = 8;   //capacity of buffer

  /* puts a character into the buffer
  *  blocks calling thread when the buffer is full (i.e. holds N characters)
  */
  public void put(char ch) throws InterruptedException;

  /* blocks calling thread until the buffer is full
  *  then returns entire contents of the buffer
  */
  public char[] getAll() throws InterruptedException;

}
```

Specify the abstract behaviour of the buffer as an *FSP* process **BUFFER** with the alphabet **{put, getAll}**.

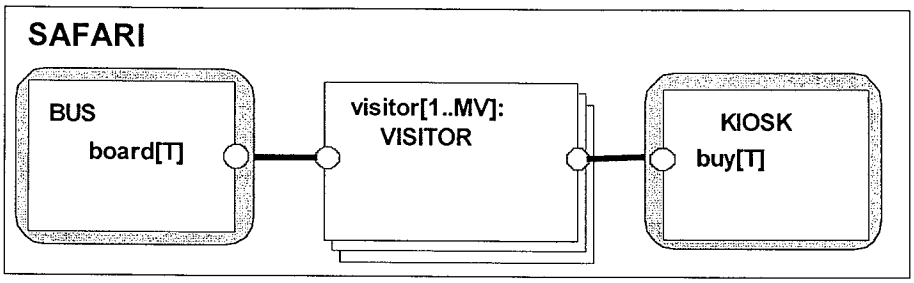
c List the program for a class that implements the **Buffer** interface. Note that your Java syntax need not be perfect.

d Extend the **BUFFER** model you specified in part b to add the action **gettwo** which models a method that gets two characters from the buffer and blocks when the buffer has less than two characters in it.

*The four parts carry, respectively, 20%, 30%, 40%, 10% of the marks.*

3a Briefly outline the two different ways of creating a new thread in Java.

b Visitors to a Safari Park buy a numbered ticket from a kiosk. Tickets are numbered in the range **T = 1..MT**. When ticket **MT** has been issued, the next ticket to be issued will be ticket numbered 1, i.e. the ticket vendor installs a new ticket roll. After buying a ticket, the visitor proceeds to the bus stop to board the bus that tours the park. The visitor may only board a bus when his/her ticket number appears on a large display. The structure diagram for a model of this system with **MV** visitors is shown below:




Given that the behaviour of VISITOR is defined by:

```
VISITOR = (buy[t:T] -> board[t] -> STOP).
```

specify the behaviour of each of the processes (BUS, KIOSK) and the composite process SAFARI in FSP.

c Implement the specifications for each of the model entities (BUS, KIOSK, VISITOR) in Java. Include the definition of a method `void build(int MV)` which creates the objects required for SAFARI.

Briefly justify your use of `notify` or `notifyAll` where used.

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

4a Explain what is meant by *transparency* of labelled transition systems in the context of safety properties for concurrent programs.

b A university department of computing consists of a systems research group and a theory research group. Due to constraints on space, these groups share a meeting room. Due to mutual antagonism between the groups, to avoid conflict, the head of the department has decreed that the meeting room can contain only systems group members or theory group members but not both at the same time. Given the following definitions:

```
const M    = 3        //maximum number of people allowed in meeting room
const Max = 7
set Hackers    = {hack[1..Max]}  //members of the systems group
set Eggheads   = {egg [1..Max]}  //members of the theory group

ACADEMIC =      (enter -> meet -> exit -> ACADEMIC).
||DEPARTMENT = (Hackers:ACADEMIC || Eggheads:ACADEMIC).
```

Specify a process **MEETINGROOM** in *FSP* that ensures that a maximum of $M$ people are allowed into the meeting room at any one time and that the meeting room cannot be occupied by both hackers and eggheads at the same time.

c Specify the following safety properties in *FSP*:

i) **NOCONFLICT** checks that the meeting room is occupied either by hackers or eggheads, but not both simultaneously.

ii) **OVERFLOW** checks that more than $M$ people do not occupy the meeting room at the same time.

Give the *FSP* composition for the system that combines the department, meeting room and the safety properties.

d Specify two progress properties in *FSP* that check, respectively, that hackers eventually get to use the meeting room and that the eggheads eventually get to use the meeting room. Give the specification for a system that models the situation in which there is a heavy demand for the meeting room. Would your progress properties be violated in this system?

*The four parts carry, respectively, 10%, 30%, 40%, 20% of the marks*