UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2000

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
BEng Honours Degree in Mathematics and Computer Science Part II
MEng 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

Tuesday 2 May 2000, 14:00
Duration: 90 minutes
(Reading time 5 minutes)

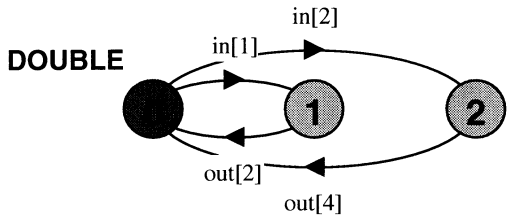
Answer THREE questions

Paper contains 4 questions

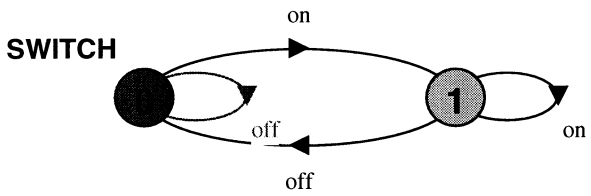
1a Define the meaning of action prefix ("->") and choice ("|") in the Finite State processes (*FSP*) notation.

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

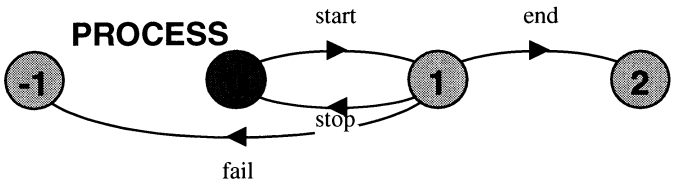
i)



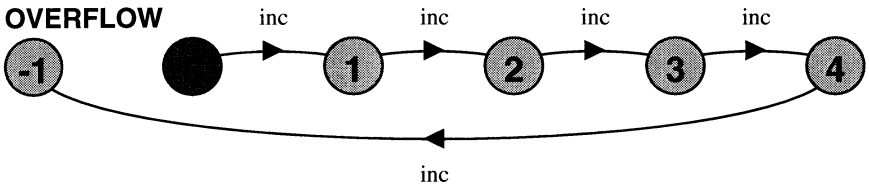
ii)



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)
```
CYCLE(N=4) = S[0],
S[i:0..3]  = (out[i] -> S[(i+1)%N]).
```
// % is modulus

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

iv)
```
PERSON    = (sleep -> dream -> PERSON
           | wake -> work  -> PERSON).
INSOMNIA  = STOP + {sleep}.
||SEATTLE = (PERSON || INSOMNIA).
```
//draw LTS for SEATTLE

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

2a Briefly describe the operation of the **notify()**, **notifyAll()** and **wait()** methods used in Java for condition synchronisation in monitors.

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[] get() throws InterruptedException;

}
```

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

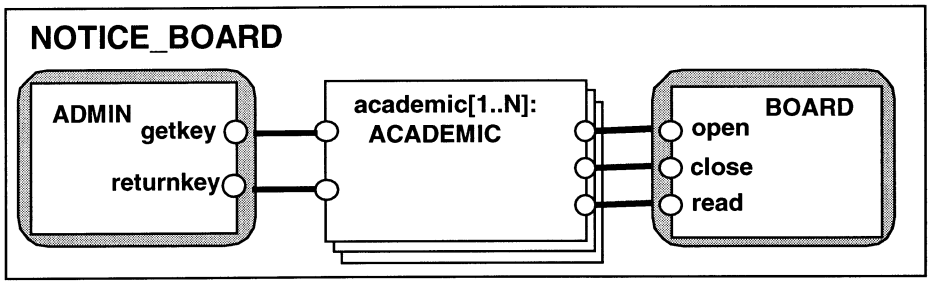
c List the program for a class that implements the **Buffer** interface.

d Extend the **BUFFER** model you specified in part b to add the action **getone** which models a method that gets a single character from the buffer and blocks when the buffer is empty.

The four parts carry, respectively, 15%, 35% , 40%, 10% of the marks

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%, 20% , 60% of the marks

4a Explain what is meant by *safety* and *liveness* properties with respect to concurrent programs.

b An academic department, due to government cuts, can only afford a single bathroom that can hold a maximum of *MP* people. The bathroom can be used by both men and women, but not at the same time. Given the following definitions:

```
const BM   = 2        //maximum number of people allowed in bathroom
const Max  = 7
set Men    = {man[1..Max]}     //set of all men in the department
set Women  = {woman[1..Max]}   //set of all women in the department

PERSON = (enter -> wash -> exit -> PERSON).
||PEOPLE = (Men:PERSON || Women:PERSON).
```

Specify a process **BATHROOM** in *FSP* that ensures that a maximum of *BM* people are allowed into the bathroom at any one time and that the bathroom cannot be occupied by both men and women at the same time.

c Specify the follow safety properties in *FSP*:

i) **UNISEX** checks that the bathroom is occupied either by men or women, but not both simultaneously.

ii) **OVERFLOW** checks that more than *BM* people do not occupy the bathroom at the same time.

Give the *FSP* composition for the system that combines people, bathroom and the safety properties.

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

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