IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2011

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 III
MSci Honours Degree in Mathematics and Computer Science Part III
MSc 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 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 C528

CONCURRENT PROGRAMMING

Tuesday 3 May 2011, 10:00
Duration: 120 minutes

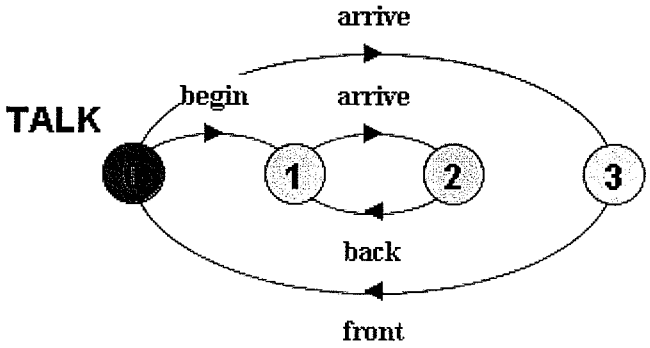
Answer THREE questions

Paper contains 4 questions
Calculators not required

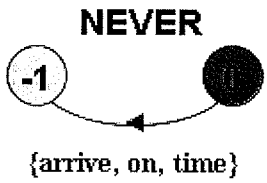
1a Explain how *non-deterministic choice* is expressed in the Finite State Processes (*FSP*) modelling notation. Briefly explain why it is useful in modeling systems.

1b For each of the following Labelled Transition Systems (*LTS*), give an equivalent *FSP* specification.

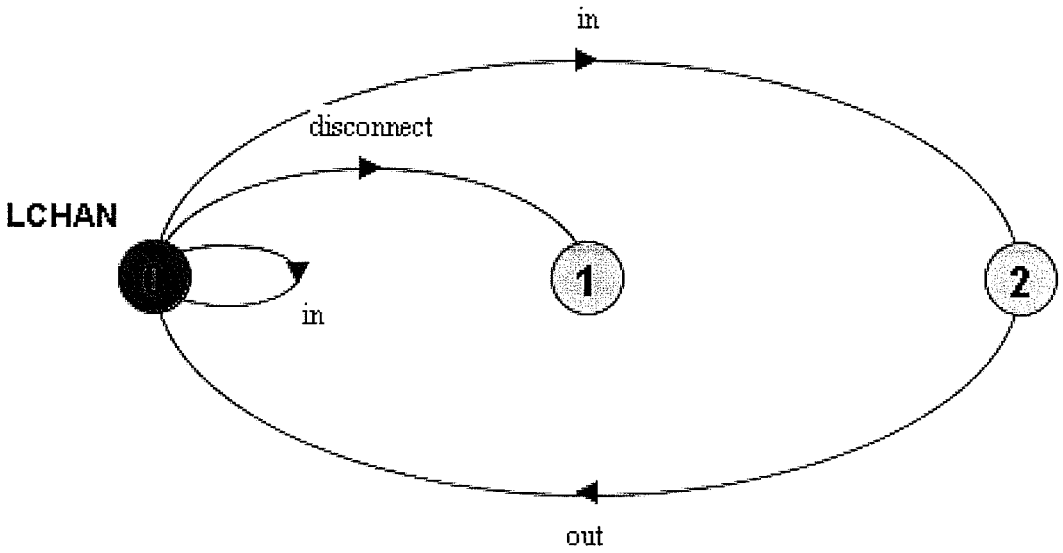
i)



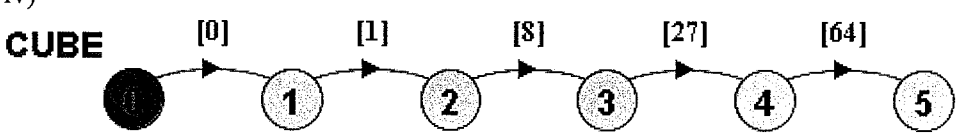
ii)



iii)



iv)



1c For each of the following *FSP* specifications, give an equivalent *LTS*.

i)
```
OVERFLOW = CAPACITY[0],
CAPACITY[i:0..3] = (drip->CAPACITY[i+1]).
```

ii)
```
BIKE = (start->(stop->STOP | faster->slower->BIKE)).
```

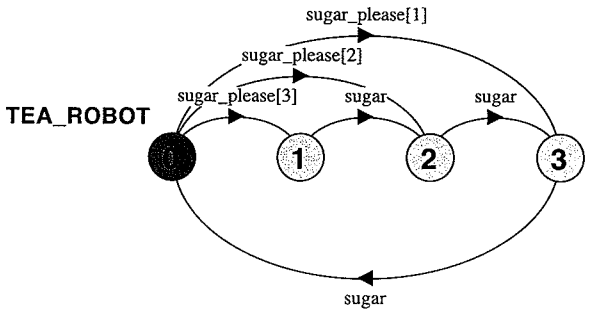
iii)
```
BLOCK = STOP + {north}.
TRAIN = (north->move->TRAIN | south->move->TRAIN).
||SYS = (BLOCK || TRAIN).          // draw LTS for SYS
```

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

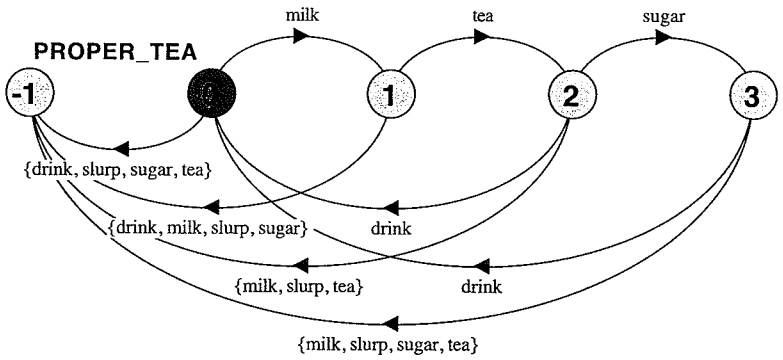
The three parts carry, respectively, 20%, 40%, 40% of the marks.

2a For each of the following Labelled Transition Systems (*LTS*), give an equivalent *FSP* specification.

i)



ii)



2b For each of the following *FSP* specifications, give an equivalent *LTS*. Explain the difference between i) and ii).

i)
```
SAFE = (sip ->   (tooHot -> SAFE |
                          hot-> drink -> SAFE)).
```

ii)
```
property SAFE = (sip ->  (tooHot -> SAFE |
                          hot-> drink -> SAFE)).
```

2c

i Explain the purpose of the `notify()` and `notifyAll()` methods in Java.

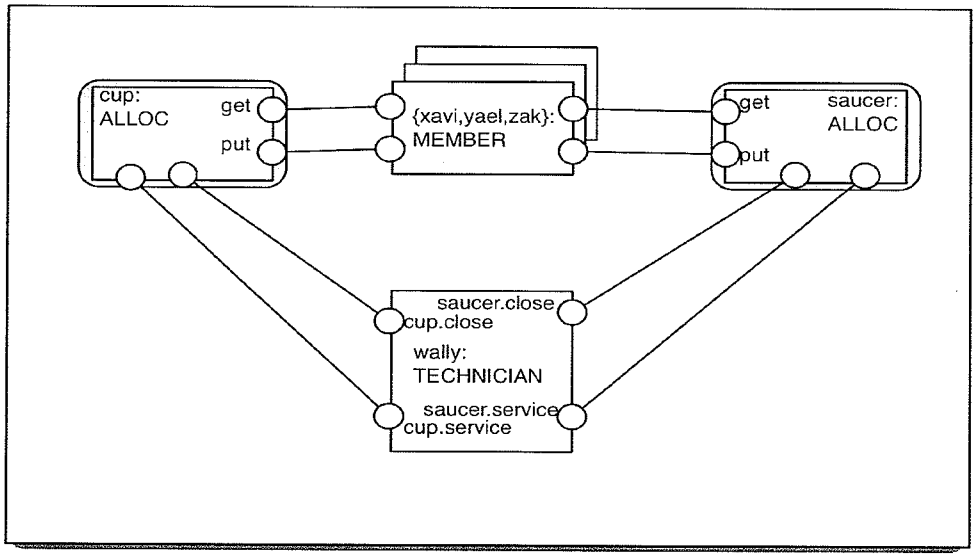
ii Describe the general situation when using `notify()` instead of `notifyAll()` may be problematic. What sort of problem might this cause?

The three parts carry, respectively, 30%, 30%, 40% of the marks.

3a In the common room used by staff members to drink tea and coffee, there is one cup allocator and one saucer allocator. When they arrive, members request a cup and saucer from the respective allocators and then serve themselves a drink. When they have finished their drink, they return cups and saucers to the allocators where they are immediately washed. A member may make repeated requests.

If sufficient cups or saucers are not available a member must wait until some are returned. To allow for maintenance of the allocator machines, the service technician closes and services each allocator. Once closed, an allocator will only accept items; it will not allocate any new ones until it has been serviced and it will not be serviced until all its items have been returned.

The structure of the system as modelled in FSP is depicted below:



Given that the behaviour of **MEMBER** and **TECHNICIAN** is defined by:

```
const MAX = 2

set All = {cup.{get,put,service,close},
           saucer.{get,put,service,close}}

MEMBER = (cup.get -> saucer.get -> drink ->
          cup.put -> saucer.put -> MEMBER) + All.

TECHNICIAN = ( cup.close -> cup.service -> saucer.close ->
               saucer.service -> TECHNICIAN) + All.
```

Specify the behaviour of the process **ALLOC** and the composite process **COMMONROOM** in *FSP*.

3b Implement the specifications of entities **MEMBER** and **ALLOC** in Java. Include the definition of a method **void build(N)** which creates the objects required for **COMMONROOM**. Note that your Java syntax need not be perfect.

3c Changing the definition of **TECHNICIAN** to:

```
TECHNICIAN = ( cup.close -> saucer.close -> cup.service ->
               saucer.service -> TECHNICIAN) + All.
```

results in a deadlock. Provide an example of how this may occur.

The three parts carry, respectively, 30%, 40% , 30% of the mark

4a Outline two ways of creating a new Thread in Java with code fragments. Briefly comment (one or two sentences) on the advantage of each.

4b Below is a Java class that offers *synchronized methods* to two resources of type **`Resource`**.

```
class MyClass {
    private Resource A a = new Resource();
    private Resource B b = new Resource();
    public synchronized void updateA(int x) {
        a.update(x);      // Update a
    }
    public synchronized void updateB(int y) {
        b.update(y);      // Update b
    }
}
```

It is now determined that the resources **`A`** and **`B`** are fully independent and are never used together. Rewrite the above class using *synchronized statements* to allow updates to **`A`** and **`B`** to be interleaved more flexibly.

4c A game that allows any N number of players, where $N > 1$, can start when all players have indicated that they are ready. Implement a *monitor* in Java called **`GameStarter`** with a method **`ready`** that ensures that all N players (assume they are implemented as separate threads) must call **`ready`** before any of them can proceed.

4d Explain a simple scenario (e.g. referring to two communicating processes, Alice and Bob) where the *nested monitor problem* may occur.

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