IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2014

BEng Honours Degree in Electronic and Information Engineering Part III
MEng Honours Degree in Electronic and Information 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

PAPER C528

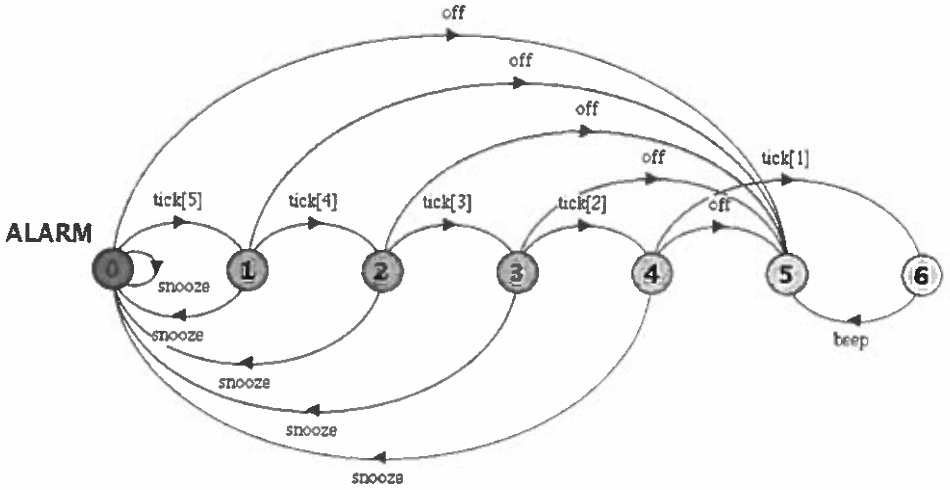
CONCURRENT PROGRAMMING

Tuesday 25 March 2014, 14:30
Duration: 120 minutes

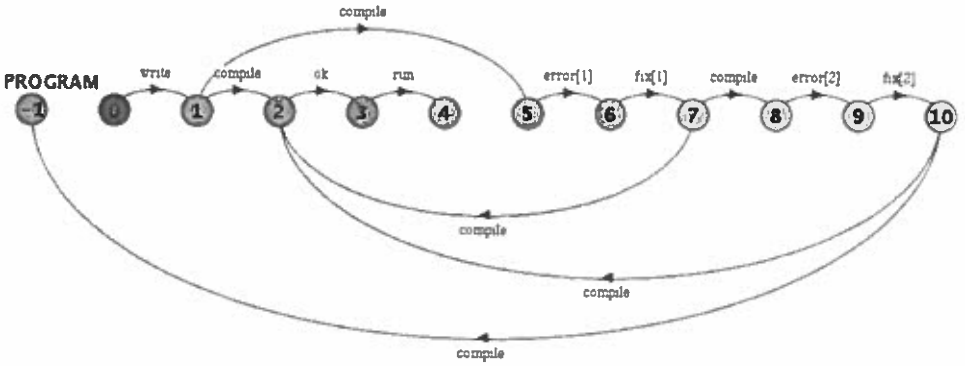
Answer THREE questions

Paper contains 4 questions
Calculators not required

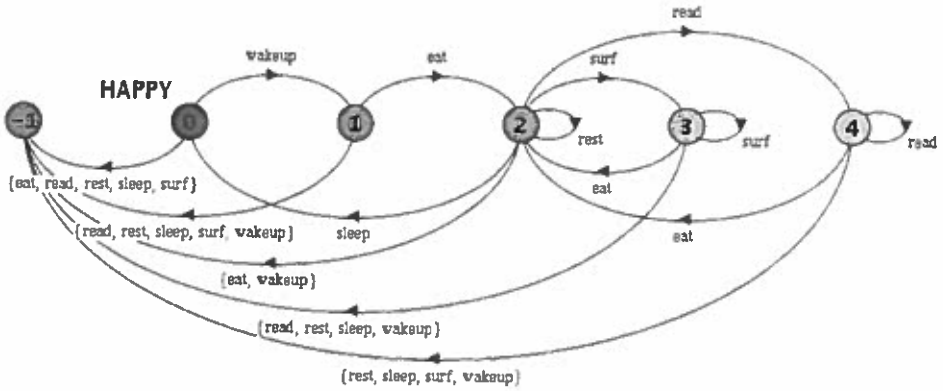
1 a For each of the following Labelled Transition Systems (LTS), give an equivalent FSP specification.



i)



ii)



iii)

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

i)
```
CONSTRAIN = STOP + {north}.
TURTLE = (north -> move -> TURTLE | south -> move -> TURTLE).
```

```
||SYS = (CONSTRAIN || TURTLE).
```

ii)
```
GATE = (in -> out -> GATE).
||AND = (a:GATE || b:GATE)/{out/{a.out, b.out}}.
```

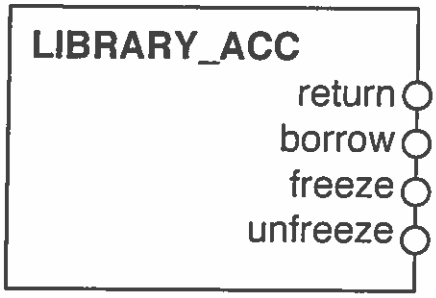
iii)
```
set A = {heads, tails}
HeadsOrTails = PICK,
PICK = (pick[i:A] -> FLIP[i]),
FLIP[i:A] = (flip[j:A] -> RESULT[i][j]),
RESULT[i:A][j:A] = (when (i==j) win -> HeadsOrTails
   |when (j!=i) lose -> HeadsOrTails).
```

c 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 the wait statement, and another thread attempts to execute `foo()`, would the "Hello World!" message be printed? Explain.

```
synchronized void foo() throws InterruptedException {
System.out.println("Hello World!");
while (cond) wait();
}
```

The three parts carry, respectively, 45%, 30%, and 25% of the marks.



2 A library has an account system in which a member is permitted to have a maximum of **M** books out on loan, i.e. it is possible to borrow books as long as the total number of books the member has on loan does not exceed this limit. Similarly, books can be returned, as long as the member has books to return. However, in some situations, the library may decide to stop the user from borrowing more books, for example, when a book has been returned damaged or has been held on loan for too long. When this is the case, the account is said to be frozen, and no further books can be borrowed until the account has been unfrozen. However, books can be returned even if the account is frozen.

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

```
const M = 10

range B = 0..M
```

`return` return a book to the library. This action is blocked if the user has no more books to return.

`borrow` borrow a book from the library. This action is blocked if the maximum number of borrowed books has been reached, or if the account is frozen.

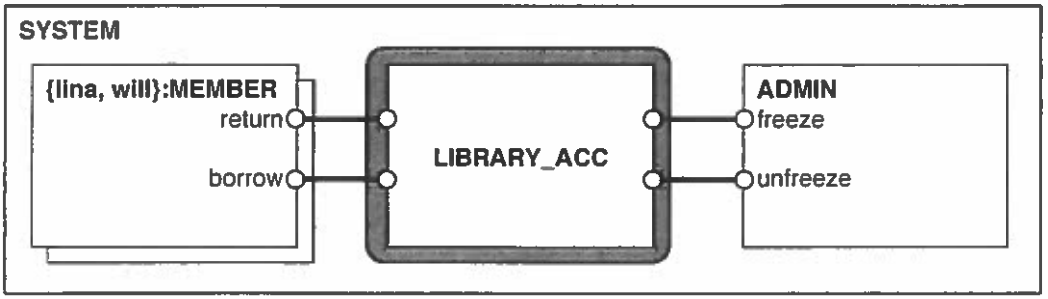
`freeze` freezes the account, preventing further borrowings.

`unfreeze` unfreezes the account, allowing borrowings.

a Specify the behaviour of LIBRARY_ACC in FSP.

b Implement the LIBRARY_ACC specification with four actions as monitor methods programmed in Java.

c Consider an extended view of the Library Account system with the **MEMBER** and **ADMIN** processes as depicted in the structure diagram below.



The behaviour of **MEMBER** and **ADMIN** are defined by:

```
MEMBER = ({borrow, return} -> MEMBER).
ADMIN = (freeze -> unfreeze -> ADMIN).
```

i) Specify the behaviour of the composite process **SYSTEM**, which includes two MEMBER processes, in FSP.

ii) Specify the safety property **NO_BORROW_WHEN_FROZEN** which ensures that no book can be borrowed if the account is frozen.

iii) Specify the progress property **ALWAYS_BORROW** which checks that it is always possible for either of the MEMBER processes to borrow books. Would this property be violated in the following composite process? Explain your answer.

```
||UNFAIR_SYSTEM = (SYSTEM) >> {will.borrow}.
```

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

3a Explain what is meant by *safety* and *liveness* in the context of concurrent programs.

b A London pub is popular with both Chelsea and Arsenal football supporters. The pub has a nicely decorated dining room, and the landlord declared that, to minimise damage, the dining room can only contain supporters of one team at any one time. Given the following definitions:

```
const M = 5
const Max = 7
set Arsenal = {arsenal[1..Max]}
set Chelsea = {chelsea[1..Max]}

SUPPORTER = (enter -> dine -> exit -> SUPPORTER).
||PUB = (Chelsea:SUPPORTER || Arsenal:SUPPORTER || DININGROOM).
```

Specify the process `DININGROOM` in FSP that ensures that a maximum of M people are allowed in the dining room at any one time, and that it cannot be occupied by both Chelsea and Arsenal supporters at the same time.

c Specify the following properties in FSP:

i) `NOCONFLICT` checks that the dining room is not occupied by both Chelsea and Arsenal fans simultaneously.

ii) `OVERFLOW` checks that not more than M people occupy the dining room at the same time.

iii) `TURNOVER` checks that, given fair choice, each fan of both teams will eventually get to eat.

d Explain how the nested monitor problem can occur in Java.

The four parts carry, respectively, 10%, 30%, 45%, and 15% of the marks.

4a Briefly explain what is alphabet extension and how it is used in modelling concurrent programs. Also provide an 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 the buffer

/**
 * Puts a character into the buffer, blocking
 * the calling thread when the buffer is full.
 */
 public void put(char ch) throws InterruptedException;
/**
 * Gets a character from the buffer. If the buffer is
 * empty, it blocks the calling thread until
 * the buffer has content.
 */
 public char get() throws InterruptedException;
/**
 * Puts a sequence of characters into the buffer, blocking
 * the calling thread until there is sufficient space
 * in the buffer.
 */
 public void putAll(char[] ch) throws InterruptedException;
/**
 * Empties the contents of the buffer, returning an empty
 * array if the buffer is empty.
 */
 public char[] getAll() throws InterruptedException;
}
```

i) Specify the abstract behaviour of the buffer as an FSP process **BUFFER** with the alphabet {`put, putAll[1..N], get, getAll`}.

ii) Write the code for a class that implements the `Buffer` interface. Note that your Java syntax need not be perfect.

The two parts carry, respectively, 20% and 80% of the marks.