Answer Question 1 and two further questions

1. Answer the following five parts

a. Describe the concept of transactions by discussing what atomicity, consistency, isolation and durability mean.

[8 marks]

Transactions are sequences of operations that are atomic, consistency preserving, isolated and durable. Atomicity means that the sequence is performed completely or not at all. Consistency preservation means that the sequence leads from one consistent state to another and inconsistencies are confined to within the transaction. Isolation means that the transaction is executed in isolation to any concurrent transactions and durability means that once the transaction is completed its changes will persist.

b. What is two-phase locking and how does it achieve which of the above transaction properties?

[5 marks]

Protocol for serializing transactions. Achieves isolation. Lock acquisition phase and lock release phase. Only resources for which lock has been acquired are accessed. Locks are only granted if request is compatible with previously granted lock requests.

c. Explain the circumstances in which you would use hierarchical locking and the lock modes that are required for this concurrency control scheme. For which situations are each of these lock modes intended?

[6 marks]

Hierarchical locking is used when there are containers that contain resources that may be locked individually. It defines the following four locking modes: A Read lock (R) indicates that a process may read a resource or any of the resource that is contained therein. A Write lock (W) indicates that a process may modify a resource or any of the resources that is contained therein. An Intention Read lock (IR) indicates that a process may have read locks on any of the resources that are contained in the IR locked resource. Finally, an Intention Write lock (IW) indicates that a process may have a write lock on the resources that are included in the container.

d. Assume you have to build the concurrency control manager of a database management system. The database consists of many database tables, each of which is composed of a set of records. Each record of the table has several attributes. You choose to use hierarchical locking. Explain what the manager needs to do for:

i. changing an attribute value; [1 marks]

request IW lock on database, table, record and W lock on attribute.

ii. reading an attribute value; [1 marks]

request IR lock on database, table and record and R lock on attribute.

iii. modifying a record; [1 marks]

request IW lock on database, table and W lock on record.

iv. reading all attributes of a record; [1 marks]

request IR lock on database, table and R lock on record.

v. updating a complete column; [1 marks]

request IW lock on database, W lock on table

vi. reading a complete column; [1 marks]

request IR lock on database, R lock on table

vii. inserting a table; [1 marks]

request W lock on database

viii. reading the complete database; [1 marks]

request R lock on database

[Subtotal 8 marks]

CONTINUED

e. Two-phase locking may lead to deadlocks. This causes many complications in the design of database management systems. Absence of deadlocks can be established using reachability analysis on LTSs. Why can this method not be used to implement deadlock-free database concurrency control?

[7 marks]

Queries and transactions are defined in an ad-hoc manner and it is more computationally expensive to compute a labelled transition system and analyse it for absence of deadlocks than it is to detect and recover from deadlocks.

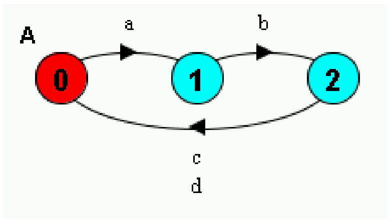
[Total 34 marks]

2. Answer the following three parts.

   a. Draw an equivalent LTS for the following FSP process definitions:

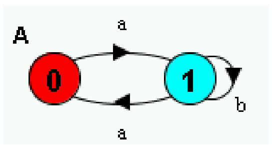
      i. `A=(a -> b -> (c -> A | d -> A)).`

      [2 marks]



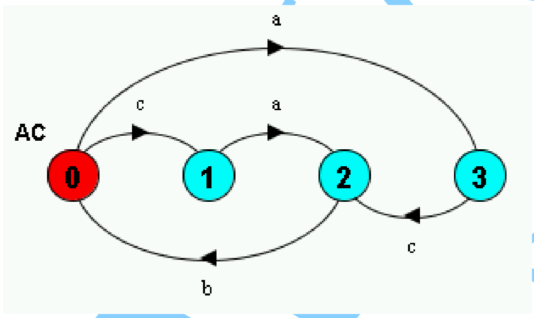
      ii. `A=(a -> B), B=(b->B | a->A).`

      [3 marks]



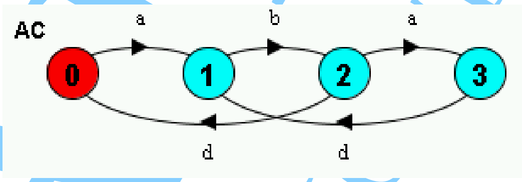
      iii. `A=(a->b->A). C=(c->b->C). ||AC=(A||C).`

      [3 marks]



      iv. `A=(a->b->A). C=(c->d->C). ||AC=(A||C)/{b/c}.`

      [3 marks]



[Subtotal 11 marks]

CONTINUED

b. You have been asked to assist in the re-engineering of the control software for the Northern Line of London Underground. There is a particular difficulty because of the intersection of the Bank/Charing Cross and the Edgware/High Barnet branches between Camden and Euston. In particular, there are trains coming from both Edgware and High Barnet that need to be dispatched onto the Bank and Charing Cross branches. For southbound trains, there is thus a piece of track that needs to be protected by signalling equipment in such a way that trains can be dispatched without crashing into each other.

Complete the following FSP model to describe the signalling equipment, taking into account that trains can concurrently arrive from the Edgware and from the High Barnet branches and must not crash into each other.

```
Signal = (green -> red -> Signal).

Train = (request -> ... -> Train).

||Branch = ...

Controller = ...

||Camden = (edgware:Branch || highbarnet:Branch || Controller).
```

[13 marks]

```
Signal = (green -> red -> Signal).
Train = (request -> green -> cross -> leave -> red -> Train).
||Branch = (Signal || Train)
Controller = (edgeware.green -> edgeware.red ->
              highbarnet.green->highbarnet.red -> Controller).
||Camden = (edgware:Branch || highbarnet:Branch || Controller).
```

c. For the model of Question 2.b specify a safety property so that you can use a model checker to prove that there is never more than one train crossing the southbound track intersection between Camden and Euston.

[9 marks]

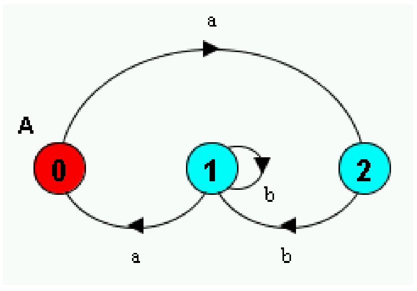
```
property SafeCrossing=(edgware.cross->edgware.leave->SafeCrossing
           |highbarnet.cross->highbarnet.leave->SafeCrossing).
```

[Total 33 marks]

3. Answer the following three parts

a. Consider the following Labelled Transition Systems and give equivalent FSP process algebras

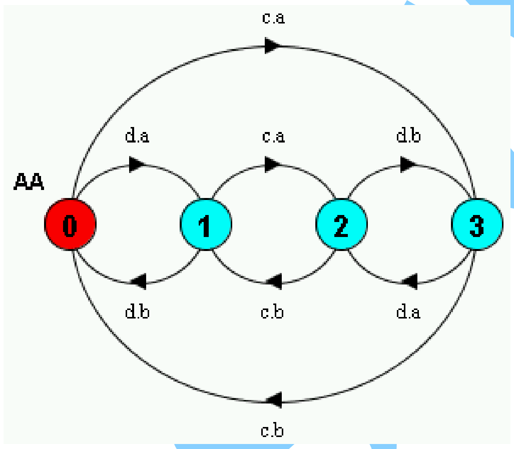
i.



[2 marks]

```
A=(a -> b -> B),
B=(b->B
  |a->A).
```

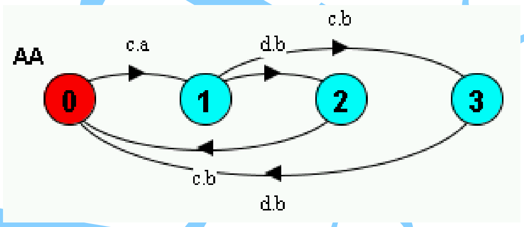
ii.



[3 marks]

```
A = (a -> b -> A).
||AA = (c:A || d:A).
```

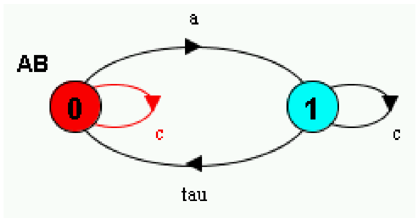
iii.



[3 marks]

```
A = (a -> b -> A).
||AA = (c:A || d:A)/{c.a/d.a}.
```

CONTINUED

iv. 

[3 marks]

```
A = (a -> aa -> A).
B = (c -> B).
||AB = (A || B)\{aa}.
```

[Subtotal 11 marks]

b. Consider the McDonald's Restaurant on the corner of Warren St and Tottenham Court Rd. At lunchtime, the restaurant has four waiters (lucy, bob, alice and jill), who take orders (mostly from UCL students). As we are in England, a student who arrives at McDonald's enters an arrival queue. She waits until one of the waiters becomes available. The waiter then takes her order, serves her and finally the student pays her bill. Model how students queue and how their orders are processed concurrently in FSP.

[13 marks]

```
const MaxIF=4
range IF=0..MaxIF
set Waiters={lucy, bob, alice, jill}

ARRIVE=ARRIVE[0],
ARRIVE[num:IF]=(enterarrq[num]->ARRIVE[(num+1)%MaxIF]).
ARRIVALQ=(enterarrq[num:IF]->ARRIVALQ[num]),
ARRIVALQ[n0:IF]=(enterarrq[num:IF]->ARRIVALQ[n0][num]
   |Waiters.order[n0]->ARRIVALQ),
ARRIVALQ[n0:IF][n1:IF]=(
    enterarrq[num:IF]->ARRIVALQ[n0][n1][num]
   |Waiters.order[n0]->ARRIVALQ[n1]),
ARRIVALQ[n0:IF][n1:IF][n2:IF]=(
    enterarrq[num:IF]->ARRIVALQ[n0][n1][n2][num]
   |Waiters.order[n0]->ARRIVALQ[n1][n2]),
ARRIVALQ[n0:IF][n1:IF][n2:IF][n3:IF]=(
    Waiters.order[n0]->ARRIVALQ[n1][n2][n3]).

Waiter = (order[c:IF] -> serve [c] -> pay[c] ->Waiter).

||McDonalds=(Waiters:Waiter|| ARRIVE || ARRIVALQ)}.
```

c. Specify the liveness property that "hungry students eventually get their lunch".

[9 marks]

```
progress LunchServed={Waiters.serve[IF]}
```

[Total 33 marks]

4. Answer the following three parts.

   a. For each of the following FSP processes, define the alphabet of:

      i. `Server = (receive -> process -> send-> Server ).` [2 marks]

         `{receive, process, send}`

      ii. `Client = (send -> wait -> receive -> Client)\{wait}.` [2 marks]

         `{receive, send}`

      iii. `{a,b}::Server` [3 marks]

         `{a.process, a.invoke, a.return  b.process, b.invoke, b.return}`

      iv.
```
||CS=(a:Client||b:Client||{a,b}::Server}/{a.send/a.invoke,
                                        b.send/b.invoke,
                                        a.receive/a.return,
                                        b.receive/b.return}.
```
[4 marks]

         `{a.process, a.send, a.receive  b.process, b.send, b.receive}`

[Subtotal 11 marks]

CONTINUED

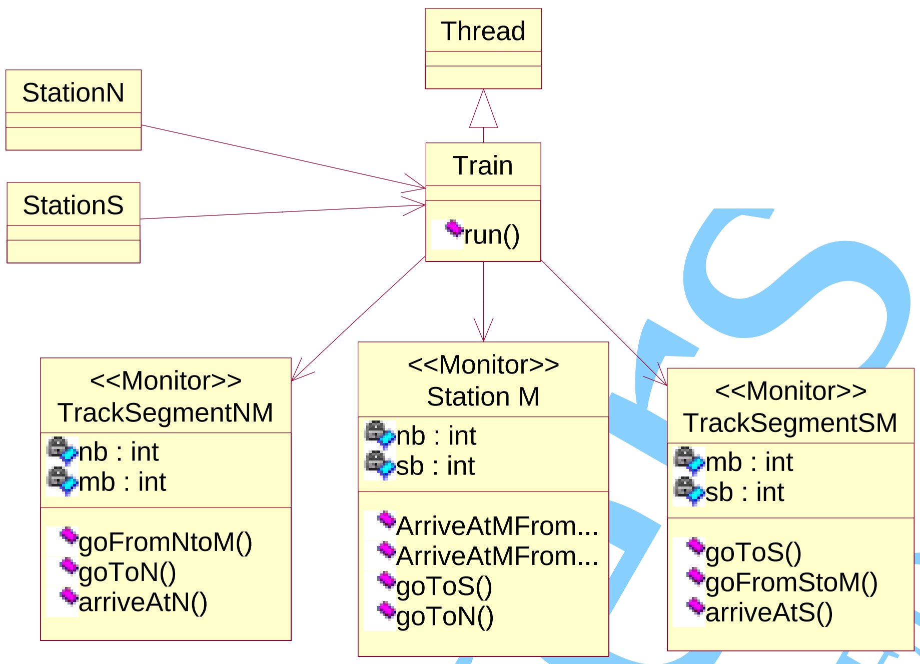
b. Consider the following FSP Process of a cross country railroad that has only a single track, which connects villages N and S via M.

```
const MaxTrains=4
range Trains=0..MaxTrains
NM=NM[0][0],
NM[nb:Trains][mb:Trains]=
  (when (nb==0 && mb<MaxTrains) go_nm->NM[nb][mb+1]
  |when (mb==0 && nb<MaxTrains) go_n->NM[nb+1][mb]
  |when (mb>0) arr_at_m_from_n->NM[nb][mb-1]
  |when (nb>0) arr_at_n->NM[nb-1][mb]
  |when (nb>0&&mb>0) unsafe->ERROR).
SM=SM[0][0],
SM[mb:Trains][sb:Trains]=
  (when (mb==0 && sb<MaxTrains) go_s->SM[mb][sb+1]
  |when (sb==0 && mb<MaxTrains) go_sm->SM[mb+1][sb]
  |when (sb>0) arr_at_s->SM[mb][sb-1]
  |when (mb>0) arr_at_m_from_s->SM[mb-1][sb]
  |when (mb>0&&sb>0) unsafe->ERROR).
N=(go_nm->N |arr_at_n->N).
S=(go_sm->S |arr_at_s->S).
M=M[0][0],
M[nb:Trains][sb:Trains]=
  (when (sb<MaxTrains) arr_at_m_from_n -> M[nb][sb+1]
  |when (nb<MaxTrains) arr_at_m_from_s -> M[nb+1][sb]
  |when (sb>0) go_s -> M[nb][sb-1]
  |when (nb>0) go_n -> M[nb-1][sb]).

||COUNTRYRR=(NM||SM||N||S||M).
```

Design the signalling control software in a UML class diagram, detailing associations, operations and attributes. Use stereotype `<<Monitor>>` to identify those classes in your design that are monitors.

[13 marks]



c. Show the implementation of the operations of the class that controls access to station M.

[9 marks]

```
class StationM {
  private
    final int MaxTrains=10;
    int nb;
    int sb;
  public synchronized void arr_at_m_from_n() {
    while(sb>=MaxTrains) wait();
    sb++;
    notifyAll();
  }
  public synchronized void arr_at_m_from_s() {
    while(nb>=MaxTrains) wait();
    nb++;
    notifyAll();
  }
  public synchronized void go_s() {
    while (sb<=0) wait();
    sb--;
    notifyAll();
  }
  public synchronized void go_n() {
    while (nb<=0) wait();
    nb--;
    notifyAll();
  }
}
```

[Total 33 marks]

CONTINUED

5. Answer the following three parts.

   a. In no more than 50 words each define the following concepts:

      i. Semaphore

      [2 marks]

      ADT with encapsulated counter and waiting list. Exports two operations signal and wait. wait checks whether counter is greater than 0. If yes decrements counter, otherwise appends calling process to waiting list. If there are waiting processes, signal wakes up first of these, otherwise it increments the counter.

      ii. Monitor

      [2 marks]

      A programming language concept that guarantees mutually exclusive access to a data structure encapsulated by the monitor.

      iii. Deadlock

      [2 marks]

      A set of processes mutually waiting for each other.

      iv. Livelock

      [2 marks]

      A process spinning while waiting for a condition that will never become true.

      v. Condition Synchronization

      [2 marks]

      Condition synchronization is implemetned inside a monitor, forces calling process to wait (outside the monitor) if conditions are not yet met. Only let them proceed when conditions are met and notifies other processes if changes to data structures have occurred.

      vi. Safety Property

      [2 marks]

      Ascertain that nothing bad will ever happen

      vii. Liveness Property

      [2 marks]

      Ascertain that something desirable will eventually happen

[Subtotal 14 marks]

b. An ftp server provides two principal operations, `put` and `get`. In order to use these operations, clients have to `open` a connection first and they `close` the connection when they no longer want to use the server. The ftp server can have several concurrent sessions with different clients. To protect ftp servers from becoming too overloaded, the number of concurrent sessions in practice has an upper boundary, which we can assume to be 4 for this exercise. Use FSP to model the behaviour of the ftp server to show how it serves a number of concurrent clients.

[11 marks]

```
const MaxSessions = 4
range Session = 0..MaxSessions
Client = (open-> Connected),
Connected = (put -> Connected
            |get -> Connected
            |close -> Client).

Server = Sessions[0],
Sessions[i:Session] = (
   when (i<MaxSessions) open -> Sessions[i+1]
  | when (i>0) close -> Sessions[i-1]
).

set Clients = {c1,c2,c3,c4,c5,c6}
||FTP = (Clients:Client || Clients::Server).
```

c. When trying to access popular ftp servers you might have experienced that it is not possible to get through. Specify the liveness property that every client will eventually be able to connect to the ftp server of Question 5.b in FSP.

[8 marks]

```
progress Download = {{Clients}.open}
```

[Total 33 marks]

END OF PAPER