Problem 1: XML

Following DTD describes XML documents containing information about components of software project. Assuming there is a document “components.xml” corresponding to this DTD.

```
<!DOCTYPE components [
    <!ELEMENT components (component+)>
    <!ELEMENT component (name, functionality, complexity, sub-component*)>
    <!ELEMENT sub-component (component, context)>
    <!ATTLIST component cid    ID    #REQUIRED>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT functionality (#PCDATA)>
    <!ELEMENT complexity (#PCDATA)>
   <!ELEMENT context (#PCDATA)>
]>
```

1) Please write a query in XPath on “components.xml” that returns the name of all components that has at least 5 sub-components.
2) Please write a query in XQuery that returns the **cid** of components that has the same complexity as the component named “merge-sort”.
3) Please design relational schema to capture the information described by this DTD.

Problem 2: B+ -Tree

1) Construct a B+-tree (n=4) for the following bulk of index entries:
   （7, 37, 67, 97, 127, 157, 187, 217）
   Assume that the tree is initially empty, and the bottom-up B+-tree construction method is used for bulk loading the above index entries.
2) For the constructed B+-tree in 1）, draw the B+ -tree after operation “insert 83”。
3) For the constructed B+-tree in 1）, draw the B+- tree after operation “delete 187”
4) Assume that the B+-tree (n=4) contains 20000 index items, please estimate the height of the tree.

Problem 3: Query Optimization

Consider following relational schema and SQL statement:

book(bid: **char(20)**, title: **char(20)**, author: **char(20)**, price: **integer**)
reader(rid: **integer**, name: **char(20)**, age: **integer**; city: char**(20)**)
borrow(rid: **integer**, bid: **char(20),** borrow_date **date,** return_date **date**)

```
select reader.rid
from reader, borrow, book
where reader.rid=borrow.rid and borrow.bid = book.bid and
        book.title =’Deep Learning’ and
        borrow_date between ’2016-01-01’ and ’2016-06-30’
```

1) Identify a relational algebra tree (or a relational algebra expression if you prefer) that reflects the order of operations that an optimizer would choose.
2) What indexes might be of help in processing this query? Explain briefly.

Problem 4: Query Cost Estimation

For the relational schema defined in **problem 4**, there are following assumptions:

- **reader** table has 1000 records. **book** table has 1000 records. **borrow** table has 79900 records.
- The ***title*** attribute of **book** is unique.
- The value of attribute ***borrow_date*** is uniformly distributed between '2012-01-01' and '2016.12.31'.
- Every reader borrowed some books. Every book was borrowed with the same chance.
- Attribute with 'integer' type needs 4 bytes. Attribute with 'date' type needs 8 bytes.
- The block size is 4K bytes.
- There are 32 blocks in buffer for performing relational operations.
- **book** table and **borrow** table both are sorted on ***bid*** attribute.

1) Estimate the size (i.e. number of records) returned by the SQL statement given in problem 4.
2) Estimate the number of blocks of relation **book** and **borrow** respectively.
3) Estimate the best cost for evaluating **book**⋈ **borrow** using Merge Join method. (Hint:（a）The cost is measured by the number of blocks transferred to main memory and the times to seek disk. （b）In order to get the best cost, please consider how many blocks in buffer should be allocated to relation **book** and **borrow** respectively)

Problem 5: Buffer Manager

DBMS generally buffer disk blocks in memory. Assuming that the maximum number of blocks that can be held in buffer is 4. Initially the buffer is empty. Consider the following schedule with assumption that each of the data items A, B, C, D and E lies on distinct blocks.

r1(A) w2(B) r1(C) w3(A) c1 w2(D) w3(E) c2 r4(D) w5(C) r5(B) w4(A)
where: ***ri(X)*** means transaction ***Ti*** read data ***X***.

wi(X) means transaction ***Ti*** write data ***X***.

ci means transaction ***Ti*** commits.

1) Figure out the first operation where an existing block in buffer must be dropped in order that another block can be read in.
2) Assume that we are using an LRU buffer-replacement policy. What block will be dropped from buffer at the time of the operation figured out in 1)
3) From the perspective of data consistency, can DBMS simply drop the block found in 2) at that time, forgetting its value, or must do something else before? Why?
4) When ***c1*** occurs in the schedule above, does this force out the blocks with the data item updated by ***T1***?

Problem 6: Two-Phase Locking Protocol

Consider the following schedule that is generated by the two-phase locking protocol.

r1(A) r2(B) r2 (C) w1(B) w4(C) r4(D) w5(A) w5(D) w3(D) r1(E)

Where: ***ri(X)*** means transaction ***Ti*** read data ***X***.

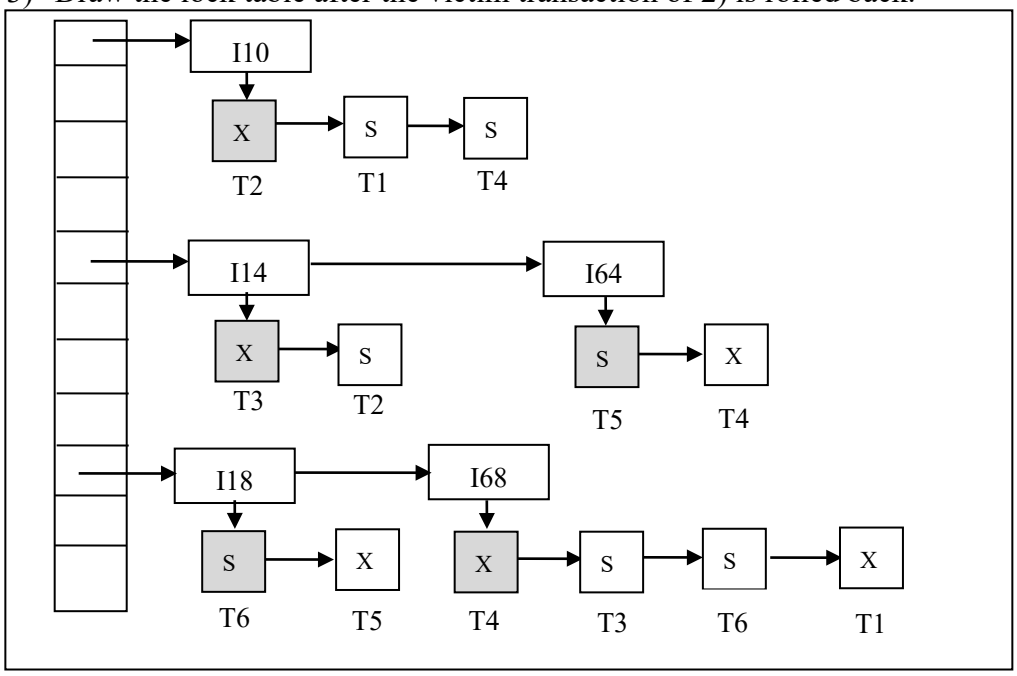
wi(X) means transaction ***Ti*** write data ***X***.

1) Draw the precedence graph of the schedule.
2) In the schedule, which transaction is the first to obtain its final lock? Which transaction is the last to obtain its final lock?
3) Is the schedule generated by the strict two-phase locking protocol? Explain briefly.

Problem 7: Lock Manager

The following figure shows an instance of lock table. There are 6 transactions (T1-T6) and 5 data items (I10, I14, I64, I18, I68). Granted locks are filled (black) rectangles, while waiting requests are empty rectangles. The lock mode(**S** or **X**) is marked in the rectangles.

1) Which transactions are involved in deadlock?
2) In order to break the deadlock, which transaction (victim) should be rolled back?
3) Draw the lock table after the victim transaction of 2) is rolled back.



Problem 8: Aries Recovery Method

Aries recovery method is used in a DBMS. Following figure is a log file just after the DBMS crashes. The log file consists of 14 log records with LSN from 6001 to 6014. The PrevLSN and UndoNextLSN values in the log records are not shown in the figure. Assuming that the last completed checkpoint is the log record with LSN 6008. Please answers following questions about the ***Aries*** recovery process.

1) From which log record does the Analysis Pass start?
2) When the Analysis Pass is complete, what is the content of the DirtyPageTable?
3) From which log record does the Redo Pass start?
4) Which transactions should be Undone?
5) Which log records is the start point of the Undo Pass? Which log record is the end point of the Undo Pass?

6001： <T1 begin>

6002： <T1 , 1001.1, 110, 111>

6003： <T2 begin>

6004： <T2 , 1001.2, 120, 122>

6005： <T2, 1002.1, 210, 211>

6006： <T3, 1003.1, 310, 311 >

6007： <T2 , 1003.2, 320, 322 >

6008： checkpoint

Txn	LastLSN
T1	6002
T2	6007
T3	6006

PageID	PageLSN	RecLSN
1001	6004	6002
1002	6005	6005
1003	6007	6006

6009： <T1 commit >

6010： <T3, 1004.1, 410, 411 >

6011： <T4 begin >

6012： <T4，1003.3, 330, 333>

6013： <T4，1004.2, 420, 422>

6014： <T3 commit >