1a. (T/F) (2 points) In a program, multiple statements can be grouped together as a transaction.
Answer: True

1b. (T/F) (2 points) The actions in a transaction are atomic and either they are all performed or none of them are performed.
Answer: True

1c. (T/F) (2 points) Since setting up a database connection is expensive, libraries like SQLAlchemy/ODBC often cache connections for future use.
Answer: True

1d. (T/F) (2 points) We cannot add constraints to the semi-structured data model.
Answer: False

1e. (T/F) (2 points) Sequential IO is much slower than random IO.
Answer: False

1f. (T/F) (2 points) In a B+ tree of maximum fanout 100, it is possible for an internal node (i.e., one that is neither the root nor a leaf) to have exactly 32 children.
Answer: False. 32 children would be way below half-full.

1g. (T/F) (2 points) For a range query, it is always better to use an index-based plan than a scan-based plan.
Answer: False. If the index is not clustered, the index-based plan would introduce random I/Os and hence may be slower than a scan-based plan.

1h. (T/F) (2 points) consider a table $R(A,B)$ with 5000 rows, where B is a unique key but not the primary key. Suppose that each B+-tree index block can hold up to 9 keys and 10 pointers. The **minimum** number of levels needed for a B+-tree index on $R(B)$ is 4 (the root counts as a level).

Answer: True. Level 1 has at most 1 node (the root). Level 2 has at most 10 nodes. Level 3 has at most 100 nodes, which can point to at most 900 rows (if the tree has 3 levels). Level 4 has at most 1000 nodes, which can point to up to 9000 rows and is enough.

1i. (T/F) (2 points) Consider the following two XPath queries:

- `//A[B/C = "foo" and B/D = "bar"]`
- `//A[B[C = "foo" and D = "bar"]]`

Every element returned by the first query will be returned by the second.
Answer:
False. Consider the following XML document:

```
<A>
<B><C>foo</C></B>
<B><D>bar</D></B>
</A>
```

The first query will return this element; the second query will not.

1j. (T/F) (2 points) consider the XPath queries above, every element returned by the second query will be returned by the first.
Answer: True. The C and D elements that make an A element satisfy the condition in the second query will make the same A element satisfy the condition in the first query

2. (Semi-structured data) Consider a course registration XML document:

```
<Registration>
  <Course capacity='140'>
    <Number>CMPT 354</Number>
    <Student><Name>Abby</Name><Grade>98</Grade></Student>
    <Student><Name>Burnie</Name><Grade>75</Grade></Student>
  </Course>
  <Course capacity='50'>
    <Number>CMPT 459</Number>
    <Student><Name>Colin</Name><Grade>90</Grade></Student>
    <Student><Name>Demi</Name><Grade>100</Grade></Student>
  </Course>
</Registration>
```

2a (4 points) Write an XPath expression that are equivalent to the XQuery below.

```
for $c in /Registration/Course
return
  if (exists($c/Student[Grade >= 90 and Grade < 95])) then $c/Number
```

Answer:

```
/Registration/Course[Student[Grade >= 90 and Grade < 95]]/Number
```

Or

```
/Registration/Course[Student[Grade >= 90][Grade < 95]]/Number
```

Or

```
/Registration/Course[count(./Student[Grade >= 90 and Grade < 95]) > 0]/Number
```

Note that `/Registration/Course[./Student/Grade >= 90 and ./Student/Grade < 95]/Number`
is wrong because there can be multiple students in a course, and the condition checks if at least one student's grade is >= 90 and at least one student's grade is < 95.
You may try XPath queries on https://codebeautify.org/Xpath-Tester

2b. (4 points) Describe what this XPath returns in English

```
/Registration/Course[Number[contains(., 'CMPT 4')] and count(Student[Grade < 80]) = 0]
```

CMPT courses where the number begins with 4 and there are no students with grade lower than 80.

2c (9 points) Consider the MongoDB database storing the same course registration info as the XML document in 2a.

```
[{'capacity':140,
  'number': 'CMPT 354',
  'students': [
    {'name': 'Abby', 'grade': 98},
    {'name': 'Burnie', 'grade': 75}
  ]},
 {'capacity':50,
  'number': 'CMPT 459',
  'students': [
    {'name': 'Colin', 'grade': 90},
    {'name': 'Demi', 'grade': 100}
  ]},
  …
]
```

Complete the MongoDB query below to retrieve the students whose grade is above the average grade for each course. Each output object has three fields, course number, student name, and student grade. Only need to write down the answers in each slot **`[Fill in]`.**

```
db.courses.aggregate([
  { [Fill in] },
  { // Group by course and calculate the average grade
    $group: {
      [Fill in],
      averageGrade: { $avg: "$students.grade" },
      students: {[Fill in]}
    }
  },
  { [Fill in] },
  { // compare student grade with the course average, $gt is >
    [Fill in]: {
      $expr: { $gt: ["$students.grade", "$averageGrade"] }
    }
  },
  {
    [Fill in]: {
      _id: 0,
      courseNumber: "$_id",
      studentName: "$students.name",
      studentGrade: "$students.grade"
    }
  }
]);
```

Answer:

```
db.collection.aggregate([
  {
    $unwind: "$students"
  },
  { // Group by course and calculate the average grade
    $group: {
      _id: "$number",
      averageGrade: {
        $avg: "$students.grade"
      },
      students: {
        $push: "$students"
      }
    }
  },
  {
    $unwind: "$students"
  },
  { // compare student grade with the course average, $gt is >
    $match: {
      $expr: { $gt: ["$students.grade", "$averageGrade"] }
    }
  },
  {
    $project: {
      _id: 0,
      course: "$_id",
      studentName: "$students.name",
      studentGrade: "$students.grade",
    }
  }
])
```

Problem 3 (Transactions). Consider the following schedule involving three transactions T1, T2, T3 (r1(A) means T1 reads A etc.)
r1(A), w1(A), r3(B), w2(A), w3(B), w1(B)

3a (3 points). Draw the precedence graph.

Answer: any graph representing T3 -> T1 -> T2

3b. (T/F, 2 points) The schedule is conflict serializable.
Answer: T

3c: (T/F, 2 points) The schedule is possible under two-phase locking (2PL).
Answer: F
T1 releases lock on A after w1(A), because T2 needs it; and T3 acquires lock on B before r3(B), then T1 has to acquire lock on B after w3(B), which is not allowed in 2PL.
If T1 retains the lock on A until it is done with w1(B), w2(A) cannot execute.

3d: (T/F, 2 points) The schedule avoids cascading rollback.
Answer:T
No dirty read from data written by other uncommitted transaction

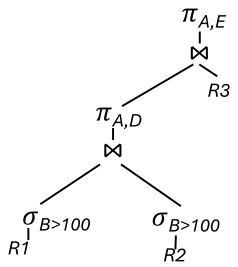
3e: (T/F, 2 points) The schedule has two equivalent serial schedules.
False:
from the precedence graph, there is only one.

Problem 4. (Indexing and Query Optimization) Given three relations R1(A,B,C), R2(B,C,D) and R3(D,E), consider following SQL Query:

```
SELECT A, E
FROM R1 NATURAL JOIN R2 NATURAL JOIN R3
WHERE R2.B > 100;
```

4a (5 points): Draw the logical plan of this query and try pushing down selections and projections as much as possible.
Note: You only need to draw the final optimized logical plan.

Answer:



4b,4c,4d (2 points each): Consider we have the following index:

```
CREATE INDEX idx on R2(B, C);
```

For each of the following SQL queries, decide whether the index idx will (A) may speed it up, or (B) will not affect its performance (the same as without using it), or (C) will slow it down.

4b: (A, B, or C): `SELECT * FROM R2 WHERE D = '001'`
Anwser: B

4c: (A, B, or C) `INSERT INTO R2 VALUES (200, 'Bart', '002')`
Anwser: C

4d: (A, B, or C) `SELECT * FROM R2 WHERE C LIKE 'Bart %'`
Anwser: A (it is possible when column B has very low cardinality)

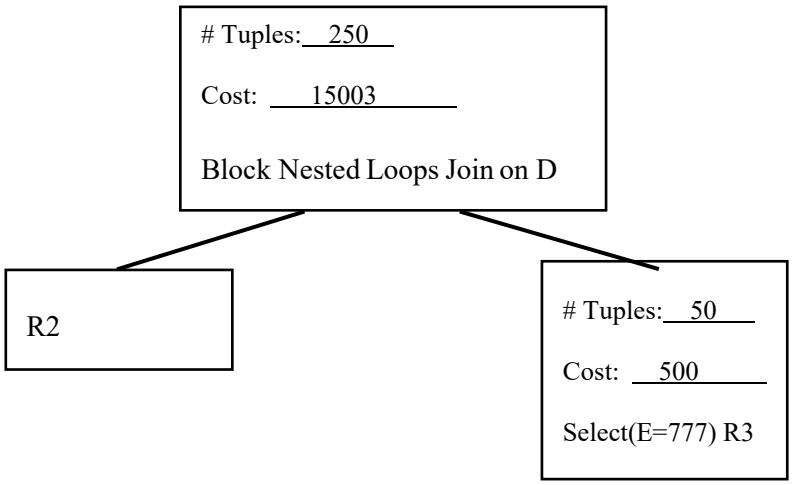
4e (8 points): Assume that we have **no indexes and 3 memory blocks**, and our only join algorithm is a block nested loops join. We execute the following query:

```
SELECT  *
FROM R2, R3
WHERE R2.D = R3.D AND R3.E = 777;
```

The database optimizer decides to execute the query with the following query plan. For each table and operator, estimate the number of output tuples and the I/O cost of each operator in the plan. Assume that all tuples are uniformly distributed, R2.C and R3.D are primary keys R2.D is a foreign key referencing R3.D, and the range of R3.E is [701, 900].
The tables have the following number of rows and blocks on disk:

Table	# Tuples	# Blocks
R2	50000	5000
R3	10000	500

Answer:



The 50 tuples from the right branch fit into 3 blocks. Each R3 tuple can join with 5 R2 tuples (given the key-foreign key and uniform distribution assumption), so the final result has 250 tuples.
The IO cost of join is B(right branch) + B(right branch) * B(left branch) = 3 + 3*5000 = 15003

4f (3 points): Now suppose R2 occupies 10000 blocks but we don't know the number of block for R3 (denoted by B(R3)), and we want to compute

```
SELECT  * FROM R2, R3 WHERE R2.D = R3.D
```

Suppose now we have 1002 memory blocks available for query processing. Which one of the following comes **the closest to the maximum value** of B(R3) for which a two-pass sort-merge join is feasible?

(A) 10^4 (B) 10^6 (C) 10^8 (D) 10^{10}

Answer: B
To accommodate one block from each run, we need ⌈10000/1002⌉ + ⌈B(R3)/1002⌉ blocks, so roughly sqrt(10000+B(R3)) < 1002.

4g (3 points): Continue with 3f, but now we want to run a two-pass hash join. Which one of the following comes **the closest to the maximum value** of B(R3) for which a two-pass hash join is feasible?

(A) 10^4 (B) 10^6 (C) 10^8 (D) 10^{10}

Answer: D
One partition of R2 is ⌈10000/1001⌉ = 10 blocks < 1002 blocks, and it already fits in memory for the probe/join pass. Therefore, R3 can be as large as possible.

Problem 5. (RA/SQL database) Consider a database storing information about coding competitions on database queries hosted by SFU in the following schema:
Student*(sid, sname, dept)*
Contest*(cid, host_dept, starttime, endtime)*
Participate*(sid, cid, num_questions_solved, rank)*
Note that:

- *Students have unique sid.*
- *Contests* have unique *cid. Host_dept* is the name of the department that provide questions. *Starttime* should be no later than *endtime*.
- Participants solve some number of questions (num_questions_solved is a non-negative integer). Rank is optional, and the value of rank (if present) is an integer >= 1. Also, *sid* and *cid* are foreign keys referring to ***Student*** and ***Contest***, respectively.

5a) (5 points) Write an **SQL statement** to create the ***Participate*** table. Include all constraints you find necessary.
Answer:

```
create table Participate(
     sid integer UNIQUE NOT NULL REFERENCES(Student(sid)),
     cid integer UNIQUE NOT NULL REFERENCES(Contest(sid)),
  num_questions_solved INTEGER NOT NULL,
  rank INTEGER,
  check (num_questions_solved >= 0 AND (rank is NULL or rank > 0)));
```

5b) (10 points) Write an **SQL query** to find all contests (*cid*) where all top-3 participants are from the same department. You can assume that there are no more than three top-3 participants (no ties).
Answer:

```
SELECT C.cid
FROM Student S, Participate P, Contest C
WHERE S.sid = P.sid AND P.cid = C.cid AND P.rank in (1,2,3)
GROUP BY C.cid
HAVING COUNT(DISTINCT S.dept) = 1

Another solution using join on S1P1, S2P2, S3P3 is OK
```

5c) (4 points) Describe succinctly what the following relational algebra query returns in English.

$$\pi_{sname,dept}\left(Student \bowtie \rho_{p1}(\sigma_{rank=1}Participate) \bowtie_{p1.sid=p2.sid\ AND\ p1.num\_problem\_solved < p2.num\_problem\_solved} \rho_{p2}(\sigma_{rank>1}Participate)\right)$$

Answer:
Find the name and department of students who won (ranked 1st in) one contest but solved fewer questions than in another contest they did not win.

5d) (5 points) Read the trigger declaration below and state what the constraint this trigger enforces.

```
CREATE FUNCTION TF_check() RETURNS TRIGGER AS $$
BEGIN
  IF TG_TABLE_NAME = 'participate' THEN
    IF EXISTS (SELECT 1 FROM Participate as p
         WHERE p.sid <> NEW.sid AND p.cid = NEW.cid
         AND ((p.num_questions_solved < NEW.num_questions_solved
               AND p.rank < NEW.rank) OR
              (p.num_questions_solved > NEW.num_questions_solved
               AND p.rank > NEW.rank)) THEN
      RAISE EXCEPTION 'constraint violation';
    END IF;
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER TG1
BEFORE INSERT OR UPDATE ON participate
FOR EACH ROW
  EXECUTE PROCEDURE TF_check();
```

Answer:
In the same contest, there are no student pairs <s1, s2> where s1's rank is better than s2 but s1 solves fewer questions than s2 does.

5e) (3 points) Assume there is a trigger on Contest and a trigger on Participate (not the one in 5d), can an UPDATE statement on Contest cause the trigger on Participate to fire? (Y or N)

Answer: Y
The trigger on Contest's action may modify Participate.