

CSE344 Final Exam

Fall 2016

December 12, 2016

- Please read all instructions (including these) carefully.
- **This is a closed-book exam. You are allowed two pages of note sheets.**
- Write your name and UW student number below.
- No electronic devices are allowed, including **cell phones** used merely as watches. Silence your cell phones and place them in your bag.
- Solutions will be graded on correctness and **clarity**. Each problem has a relatively simple solution. Partial solutions will be graded for partial credit.
- There are 20 pages in this exam, not including this one.
- There are 6 questions, each with multiple parts. If you get stuck on a question move on and come back to it later.
- You have 110 minutes to work on the exam.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. **Do not** use any additional scratch paper.
- Relax. You are here to learn. Good luck!

By writing your name below, you certify that you have not received any unpermitted aid for this exam, and that you will not disclose the contents of the exam to anyone in the class who has not taken it.

NAME: _____

STUDENT NUMBER: _____

| Problem | Points |
|----------------|---------------|
| 1 | 15 |
| 2 | 36 |
| 3 | 32 |
| 4 | 48 |
| 5 | 44 |
| 6 | 25 |
| Total | 200 |

Problem 1: Warm up (15 points total)

Select either True or False for each of the following questions. For each question you get 1 point for answering it correctly, -0.5 point for an incorrect answer, and 0 point for no answer. The minimum you will get for this entire problem is 0.

1. Given $R(A,B)$, `SELECT A+B FROM R` is a monotonic query.
True False

2. Every superkey is also a key.
True False

3. Spark disallows RDDs to be nested within other RDDs.
True False

4. A query plan that uses broadcast joins can have lower cost than one that uses partition hash joins for the same query.
True False

5. BCNF is neither the third nor the fourth normal form.
True False

6. Setting isolation level to read committed can avoid the phantom problem.
True False

7. MapReduce promotes fault tolerance by writing intermediate files to disk.
True False

8. All serializable schedules are conflict-serializable.
True False

9. When doing a broadcast join, we typically broadcast the largest relation.
True False
10. Serial execution guarantees ACID.
True False
11. It is possible to start reducers while some mappers are still running in Hadoop.
True False
12. Strict 2PL ensures serializability of transactions.
True False
13. The conflict precedence graph can be used to find serial schedules.
True False
14. If A and B are sets, then a relation R can be viewed as a subset of $A \times B$.
True False
15. If a transaction is atomic, it either shows all the effects of its operations or none of them.
True False

Problem 2: Writing Queries (36 points total, 6 points each)

Given the following relations on individuals and their friends:

Person(id, name, age)

Friends(id1, id2) -- both attributes are foreign keys to Person

If A (id: 10) and B (id: 23) are friends, then (10, 23) and (23, 10) are both stored in Friends.

a) Write a SQL query that returns the name and the number of friends that each person has.

b) Write a SQL query that returns the friends distribution. For example, if 2 individuals have 10 friends, 3 have 4 friends, and 5 have 0 friends, then you should return:

| friends | count |
|---------|-------|
| 10 | 2 |
| 4 | 3 |
| 0 | 5 |

Note that there are no tuples for 1 friend, 2 friends, etc as the count for those entries are 0.

(Problem 2 continued)

Relations duplicated here for your reference:

Person(id, name, age)

Friends(id1, id2) -- both attributes are foreign keys to Person

c) Write a SQL query that returns the name of the oldest individual(s) in the dataset.

d) Each pair of friends are currently stored twice in the Friends relation, as noted above. Write a Datalog query that returns each pair of friends only once. You can use negation if needed.

(Problem 2 continued)

Relations duplicated here for your reference:

Person(id, name, age)

Friends(id1, id2) -- both attributes are foreign keys to Person

e) Write a SQL query to find friends of friends of Person 1 (id: 10) who Person 1 is not already friends with.

f) Write a relational calculus query that returns all people who are only friends with those who have one other friend.

Problem 3: Transactions (32 points total)

Given the following three transactions:

T1: R(A), W(B), I(D), R(C)

T2: R(B), R(D), W(C)

T3: R(D), R(C), R(D), W(A)

Assume that R(X) reads all tuples in table X, W(X) updates all tuples in X, and I(X) inserts one new tuple in X. Co and Ab mean commit and abort, respectively. In the following, indicate the **strongest** isolation level that can generate each schedule, with serializable being the strongest isolation level. If serializable, then also give the serial schedule. (4 points each)

a) R1(A); W1(B); I1(D); R3(D); R2(B); R3(C); R3(D); R2(D); R1(C); W2(C); W3(A); Co1; Co2; Co3;

None Read uncommitted Read committed Repeatable read Serializable

Serial schedule:

b) R2(B); R1(A); R3(D); R3(C); R2(D); W2(C); Co2; R3(D); W3(A); Ab3; W1(B); I1(D); R1(C); Co1;

None Read uncommitted Read committed Repeatable read Serializable

Serial schedule:

c) R1(A); R2(B); R3(D); R3(C); R2(D); I1(D); W2(C); Co2; R3(A); W1(B); W3(D); Co3; R1(C); Ab1;

None Read uncommitted Read committed Repeatable read Serializable

Serial schedule:

(Problem 3 continued)

Transactions copied here for your reference.

T1: R(A), W(B), I(D), R(C)

T2: R(B), R(D), W(C)

T3: R(D), R(C), R(D), W(A)

d) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **both shared and exclusive table locks**? If so write such a schedule with lock / unlock ops, and explain why the transactions are deadlocked. Otherwise write "No". Use L1(A) to refer to T1 locking table A, and U1(A) for unlocking. (5 points)

e) Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **only exclusive table locks**? If so write such a schedule with lock and unlock operations and indicate why the transactions are deadlocked. Otherwise write "No". Use L1(A) to refer to T1 locking table A, and U1(A) for unlocking. (5 points)

(Problem 3 continued)

Transactions copied here for your reference.

T1: R(A), W(B), I(D), R(C)

T2: R(B), R(D), W(C)

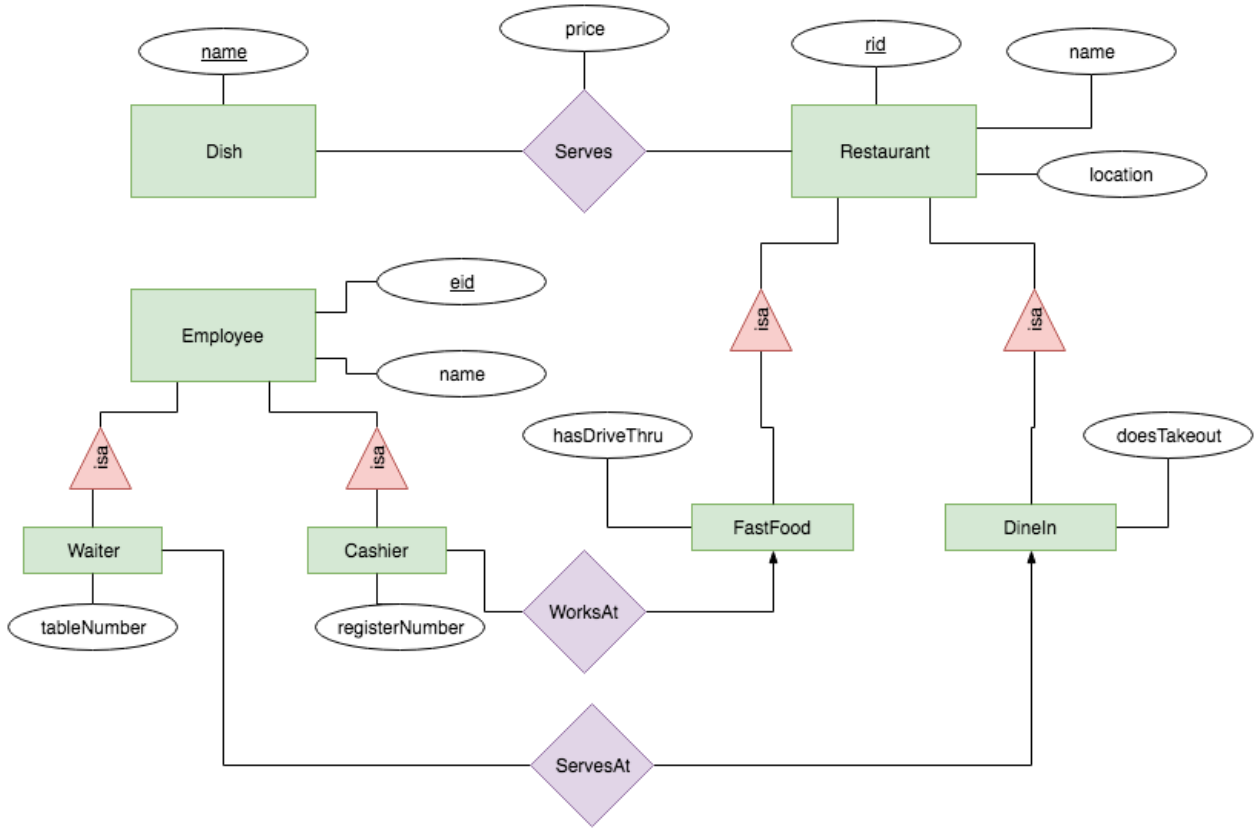
T3: R(D), R(C), R(D), W(A)

f) Does there exist a schedule of the above transactions that would result in a manifestation of the phantom problem under non-strict 2PL with exclusive table locks? If so write out such a schedule with lock and unlock operations and indicate why there is a phantom problem. Otherwise write "No". (5 points)

g) Suppose we change locking granularity to tuple rather than table level, where we only lock the tuples that are read / written / inserted from the affected table. Is there a schedule of the above transactions that would result in a manifestation of the phantom problem under non-strict 2PL with exclusive tuple locks? If so write out such a schedule with lock and unlock operations and indicate why there is a phantom problem. Otherwise write "No". (5 points)

Problem 4: Conceptual Design (48 points total)

a)



Convert the above E/R diagram to relations in BCNF form. Include all keys and foreign keys. Use the following notation and explicitly state foreign key relationships. For instance:

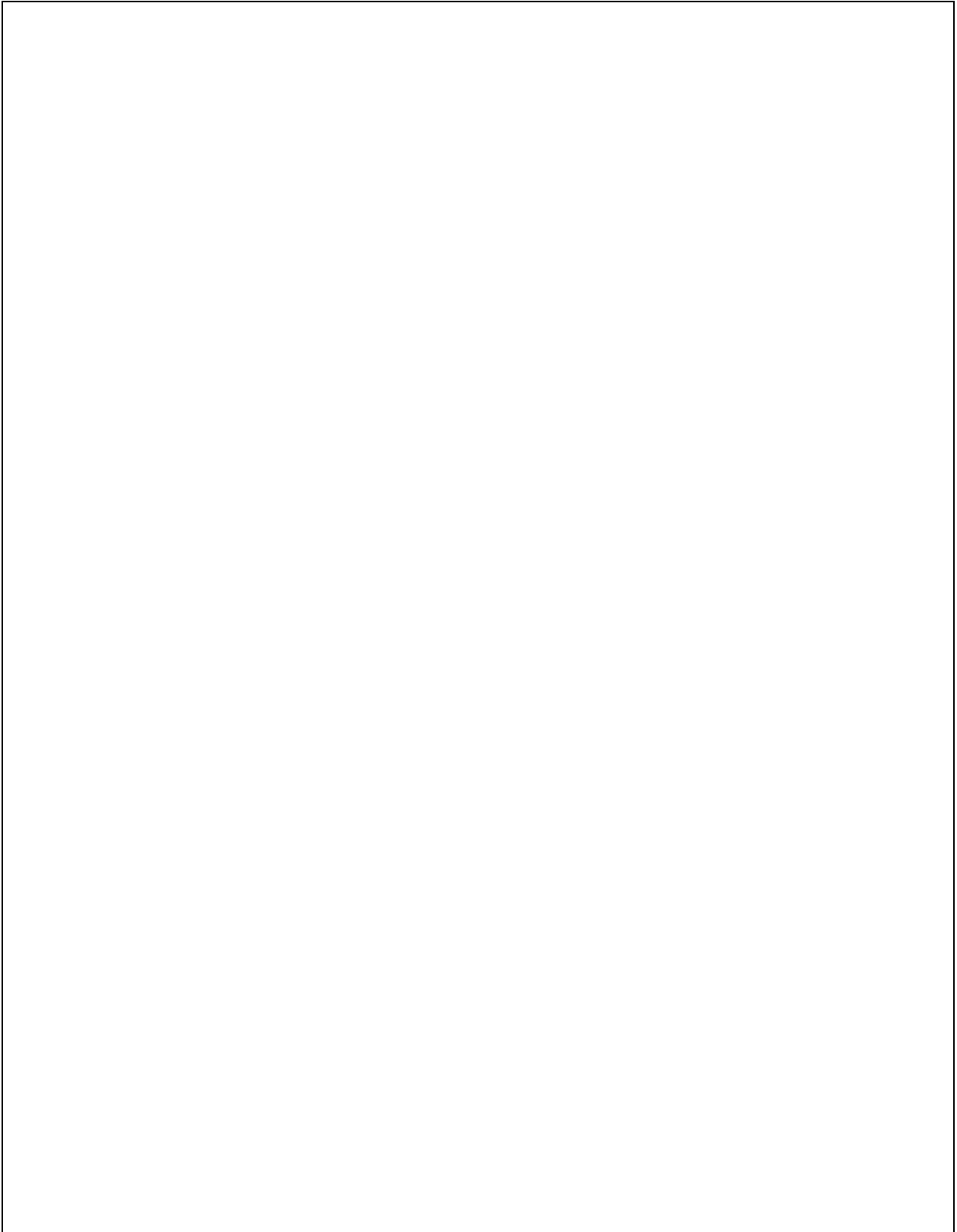
R(a, b)

S(c, d) -- c is a foreign key to R

You can ignore data types for all attributes. (33 points)

(Problem 4 continued)

a)



(Problem 4 continued)

b) Consider relation $R(A, B, C, D, E)$ with functional dependencies:

$AB \rightarrow C, C \rightarrow D, BD \rightarrow E$

Which of the following attribute sets does not functionally determine E? (3 points)

- a) AB
- b) AC
- c) BC
- d) ABC

c) Let relation $R(A, B, C, D)$ satisfy the following functional dependencies:

$A \rightarrow B, B \rightarrow C, C \rightarrow A$

Call this set S_1 . A different set S_2 of functional dependencies is *equivalent* to S_1 if exactly the same FDs follow from S_1 and S_2 . Which of the following sets of functional dependencies is equivalent to the set above? (3 points)

- a) $B \rightarrow AC, C \rightarrow AB$
- b) $A \rightarrow B, B \rightarrow A, C \rightarrow A$
- c) $A \rightarrow BC, C \rightarrow AB$
- d) $A \rightarrow BC, B \rightarrow AC, C \rightarrow AB$

d) Find all non-trivial (i.e., $A \rightarrow A$, all attributes \rightarrow all attributes) functional dependencies from the following relation: (6 points)

| A | B | C |
|----|----|----|
| a2 | b2 | c1 |
| a1 | b1 | c3 |
| a1 | b1 | c2 |
| a3 | b3 | c1 |

e) Suppose relation $R(A, B, C)$ has tuples $(0, 0, 0)$ and $(1, 2, 0)$, and it satisfies the functional dependencies $A \rightarrow B$ and $B \rightarrow C$. Which of the following tuples may be inserted into R legally? (3 points)

- a) $(0, 0, 1)$
- b) $(1, 2, 1)$
- c) $(0, 1, 1)$
- d) $(0, 0, 2)$
- e) None of the above.

Problem 5: Parallel Query Processing (44 points total)

Given the following query with relations $R(A, B)$ and $S(C)$:

```
SELECT R.A, MAX(R.B)
FROM R, S
WHERE R.A = S.C AND R.B > 10
GROUP BY R.A
```

With the following data statistics:

$T(R, a) = 10000$

$V(R, b) = 100$

$V(R, a) = 20$

$\text{Min}(R, a) = 10$; $\text{Max}(R, a) = 50$, evenly distributed.

$\text{Min}(R, b) = 20$; $\text{Max}(R, b) = 100$, evenly distributed.

$T(S, c) = 5000$

$V(S, c) = 20$

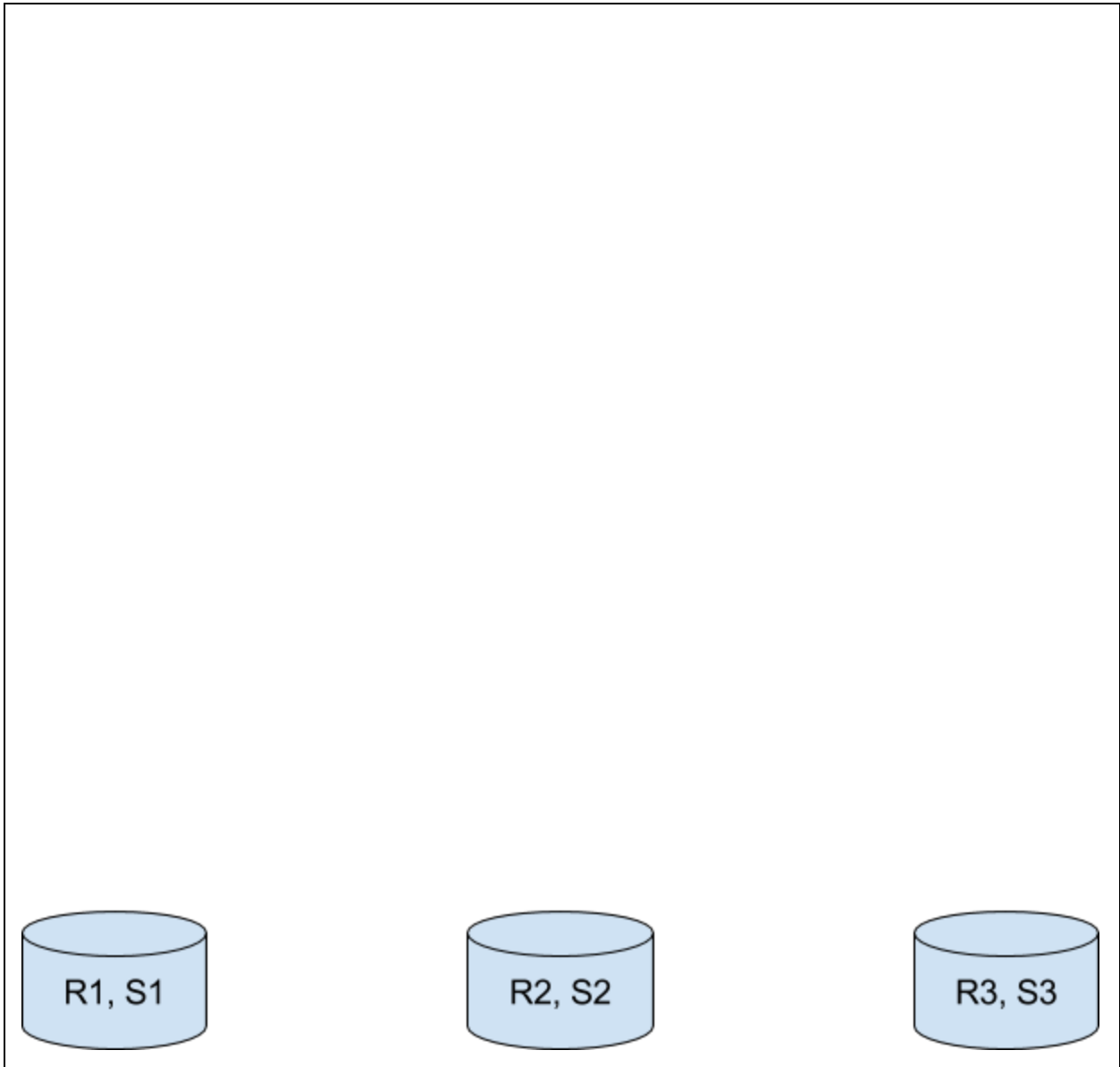
$\text{Min}(S, c) = 10$; $\text{Max}(S, c) = 50$, evenly distributed.

(problem 5, continued)

Query copied here for your reference.

```
SELECT R.A, MAX(R.B)
FROM R, S
WHERE R.A = S.C AND R.B > 10
GROUP BY R.A
```

a) Suppose that R and S are partitioned across 3 different machines using random block partitioning, and no indexes are available on any of the machines. Draw the relational algebra plan that you would use to execute the query above. Choose only between broadcast and shuffle for joins. You do not need to indicate how joins are executed locally on each machine. (17 points)



(Problem 5, continued)

b) Your friend thinks that random block partitioning is not the best partitioning scheme for this query. Are there better ways to partition R and S across the machines such that the given query can run faster? If so state how you would partition R and S, and why would doing so will lead to better performance. If not write “No”. (5 points)

c) Given a list of pixel values, encoded as (R,G,B), compute a histogram for each of the three color channels, where each pixel value can range from 0 to 255. Return the histogram as key-value pairs, with the key being a string of either “R” (red), “G”, (green), and “B” (blue), and value being an array of 255 integers, where each entry indicates the number of pixels in the input pixels that has that value of R/G/B. For instance, given this input of three pixels:

`(3, 2, 3), (4, 2, 5), (3, 1, 5) // each pixel is of the form (r, g, b)`

You should output:

```

("R", [0, 0, 0, 2, 1, 0 ... 0]) // 255 entries total in array
("G", [0, 1, 2, 0 ... 0])
("B", [0, 0, 0, 1, 0, 2, 0 ... 0])
    
```

Note that $R[0] = 0$ since none of the input pixels has red value equals to 0, but $R[3] = 2$ since two of the input pixels has red value equals to 3.

(Problem 5, continued)

Write a **single stage** map reduce program in Hadoop that computes the output above. You can assume that the input is already parsed and is given to you as a list of `Pixel` objects. You can assume that the Hadoop `emit` method handles object serialization. Use only the Hadoop constructs that we discussed in class.

Do not worry too much about getting Java syntax correct, but make sure you clearly state what key-value pairs are emitted by `map` and how they are reduced. (17 points)

```
void map (Pixel [] pixels) {
// p.r accesses the red value of pixel p as an integer, and similarly for
// p.g and p.b

}

// fill in the method signature of reduce

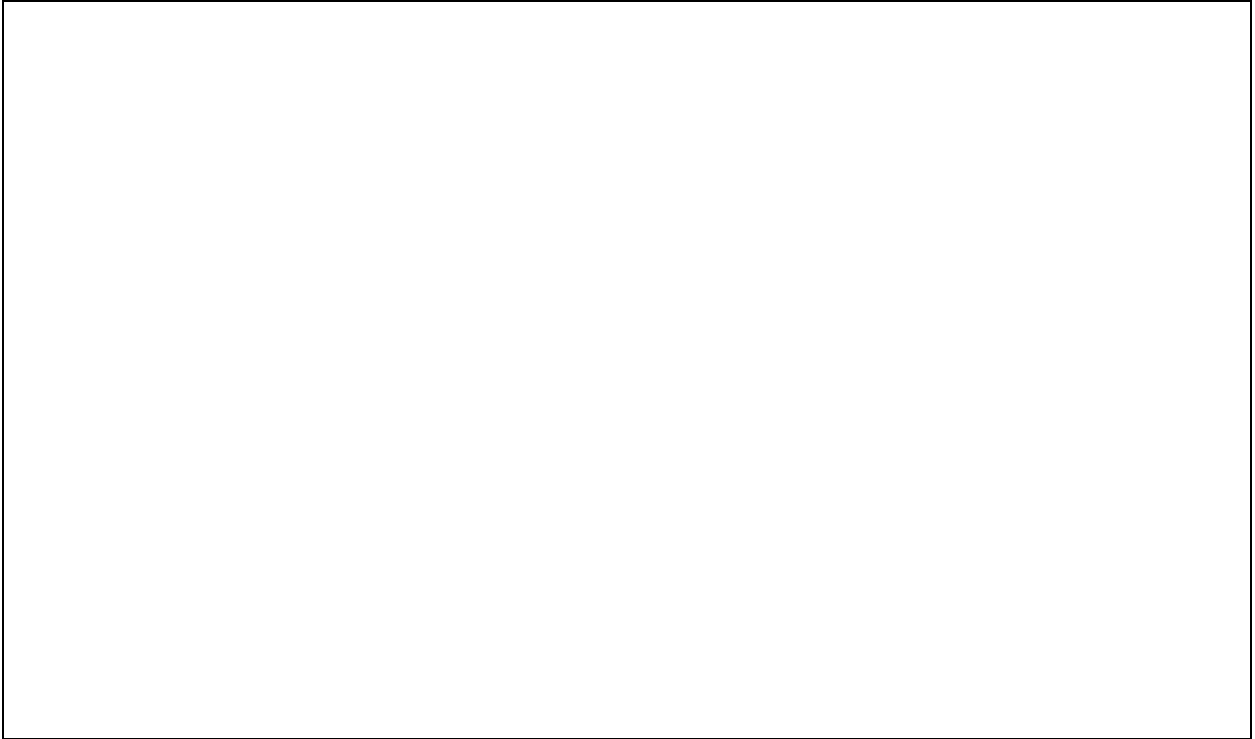
void reduce (                ) {

}

}
```

(Problem 5, continued)

d) Explain, in a few words, why your solution above performs better than the naive implementation of having the mappers broadcast the entire `pixels` array to all reducers (Obviously you will get no points for c) if that was what you wrote).
(5 points)



Problem 6: Semi-structured Data (25 points total)

Suppose we have a JSON document that encodes individuals and their children, where each individual has a unique “id” field:

```
{ "individuals":
  [ { "id": 1,
    "name": Adam,
    "age": 70,
    "children": [ { "id": 2,
      "name": John,
      "age": 49,
      "children": [ { "id": 24, "name": Liz, "age": 21 } ] } ]
    },
  { "id": 3,
    "name": Beatrice,
    "age": 35,
    "children": [ { "id": 55, "name": Peter, "age": 10 },
      { "id": 90, "name": Zoe, "age": 14 } ]
    }
  ]
}
```

Your friend plans to run transactions on this dataset to update existing fields (i.e., individuals cannot be added or removed). She decided to use locks to ensure atomicity. Her locking scheme is as follows:

- Each individual is associated with one exclusive lock.
- To read or write to the children field of individuals, the transaction first releases the lock on the parent individual, and grabs the lock on the child individual. For instance, to change Peter’s age, the transaction will first grab the lock on Beatrice, release that lock, and then grab the lock on Peter, using three operations.
- Each lock is released immediately after the corresponding operation on the individual is done.

a) Which isolation level does this locking scheme corresponds to? Explain why. (Hint: not serializable) (5 points)

(problem 6, continued)

Data duplicated here for your reference.

```
{ "individuals":  
  [ { "id": 1,  
      "name": Adam,  
      "age": 70,  
      "children": [ { "id": 2,  
                    "name": John,  
                    "age": 49,  
                    "children": [ { "id": 24, "name": Liz, "age": 21 } ] } ]  
    },  
    { "id": 3,  
      "name": Beatrice,  
      "age": 35,  
      "children": [ { "id": 55, "name": Peter, "age": 10 },  
                  { "id": 90, "name": Zoe, "age": 14 } ]  
    } ]  
}
```

b) After realizing the problem, your friend wants to keep locking at the individual level but make her locking scheme fully serializable. How would you change her locking scheme above to do so? (10 points)

(problem 6, continued)

Data duplicated here for your reference.

```
{ "individuals":
  [ { "id": 1,
      "name": Adam,
      "age": 70,
      "children": [ { "id": 2,
                     "name": John,
                     "age": 49,
                     "children": [ { "id": 24, "name": Liz, "age": 21 } ] } ]
    },
    { "id": 3,
      "name": Beatrice,
      "age": 35,
      "children": [ { "id": 55, "name": Peter, "age": 10 },
                    { "id": 90, "name": Zoe, "age": 14 } ]
    }
  ]
}
```

c) Convert the data into relations. Show both your relational schemas and how the data above will be encoded. Make sure that your schemas are in BCNF. Don't worry about data types. Use the notations below: (10 points)

Foo(a, b)

| a | b |
|----|---------|
| 10 | "hello" |
| 20 | "world" |

-- END OF EXAM --

-- Thank you for taking this class. Happy Holidays! --