CSE 414 Final Examination

June 6, 2016, 2:30pm - 4:20pm

Name: _____

Question	Points	Score
1	60	
2	30	
3	10	
4	50	
5	50	
Total:	200	

- This exam is CLOSED book and CLOSED devices.
- You are allowed TWO letter-size pages with notes (both sides).
- You have 1h:50 minutes; budget time carefully.
- Please read all questions carefully before answering them.
- Some questions are easier, others harder; if a question sounds hard, skip it and return later.
- Good luck!

1 SQL and Relational Languages

1. (60 points)

   An online picture sharing company uses a database with the following schema:

   ```
   Users(uid, uname, city)
   Picture(pid, author, size, pdf)
   ```

   - `Users` stores all users; `uid` is the key.
   - `Picture` stores their pictures; `pid` is the key; `author` is the `uid` of the picture's author; `size` represents the size of the picture in bytes; `pdf` is the actual pdf content of the picture.
   - `uid`, `pid`, `author`, `size` are integers; `uname`, `city`, `pdf` are text.

Users(uid, uname, city)
Picture(pid, author, size, pdf)

(a) (10 points) Write the SQL statements to create the tables for this database.

Solution:

```
drop table if exists Picture;
drop table if exists Users;

create table Users (
    uid int primary key,
    uname text not null,
    city text not null);

create table Picture (
     pid int primary key,
     author int not null references Users(uid),
     size int not null,
     pdf text);

insert into Users values(1,'Alice','Denver');
insert into Users values(2,'Bob','Denver');
insert into Users values(3,'Carol','Portland');
insert into Users values(4,'David','Denver');
insert into Users values(5,'Evan', 'Seattle');


insert into Picture values(10, 1, 1500000, 'abcd');
insert into Picture values(20, 1, 1500000, 'bcde');
insert into Picture values(30, 2, 1500000, 'cdef');
insert into Picture values(40, 1, 1500000, 'defg');
insert into Picture values(50, 2, 1500000, 'efgh');
insert into Picture values(60, 5,  500000, 'fghk');
insert into Picture values(70, 5, 4000000, 'ghkm');
```

2 points were taken off for mssing `primary key` or `references`.
Almost everyone answered this perfectly.

```
Users(uid, uname, city)
Picture(pid, author, size, pdf)
```

(b) (5 points) Write a SQL query that returns all users that have posted both a picture larger than 1MB (`size > 1000000`) and a picture smaller than 1MB. Your query should return the users' `uid` and `uname`

Solution:

```
select x.uid, x.uname
from users x, picture y, picture z
where x.uid = y.author and x.uid = z.author
  and y.size > 1000000 and z.size <= 1000000;
```

3 points off for nested queries

4 points off for `from user, picture`

3 points off for `from user, user, picture, picture`

Most students answered correctly.

```
Users(uid, uname, city)
Picture(pid, author, size, pdf)
```

(c) (10 points) Write a SQL query that retrieves all users who do not have any picture greater than 1MB (`size > 1000000`). Your query should return the users' `uid` and `uname`

Solution:

```
select x.uid, x.uname
from Users x
where not exists (select *
                  from Picture y
                  where y.size > 1000000 and x.uid = y.author);
```

Other correct solutison use `not in` or $<$ `ALL`

2 points off for joining `Users` with `Picture` (since that fails to return users who didn't post any pictures).

No points taken off for unnecessarily joining `Picture` with `Users`

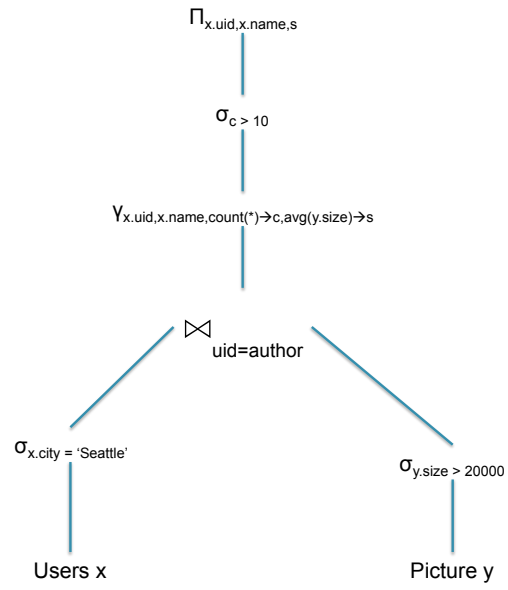
Many students answered correctly.

```
Users(uid, uname, city)
Picture(pid, author, size, pdf)
```

(d) (5 points) Write a Relational Algebra expression that is equivalent to the following SQL query:

```
select x.uid, x.uname, avg(y.size) as s
from Users x, Picture y
where x.uid = y.author
  and x.city = 'Seattle'
  and y.size > 20000
group by x.uid, x.uname
having count(*) > 10;
```

You may either write a Relational Algebra expression, or draw a query plan.



Solution:

```
Users(uid, uname, city)
Picture(pid, author, size, pdf)
```

(e) Consider the following query:

```
select x.uid, x.uname,
         (select count(*)
          from Picture y
          where x.uid = y.author and y.size > 1000000)
from Users x
where x.city = 'Denver';
```

For each query below indicate whether it is equivalent to the given query (meaning: it returns the same answers). All queries are syntactically correct.

i. (2 points) Q1:

```
select x.uid, x.uname, count(*)
from Users x, Picture y
where x.uid = y.author  and x.city = 'Denver' and y.size > 1000000
group by x.uid, x.uname;
```

i. **No**

Your answer:

ii. (2 points) Q2:

```
select x.uid, x.uname, count(y.pid)
from Users x left outer join Picture y on x.uid = y.author and y.size > 1000000
where x.city = 'Denver'
group by x.uid, x.uname, x.city;
```

ii. **Yes**

Your answer:

iii. (2 points) Q3:

```
select x.uid, x.uname, count(*)
from Users x left outer join Picture y on x.uid = y.author and y.size > 1000000
group by x.uid, x.uname, x.city
having x.city = 'Denver';
```

iii. **No**

Your answer:

iv. (2 points) Q4:

```
select x.uid, x.uname, count(y.pid)
from Users x left outer join Picture y on x.uid = y.author and y.size > 1000000
group by x.uid, x.uname, x.city
having x.city = 'Denver';
```

iv. **Yes**

Your answer:

v. (2 points) Q5:

```
select x.uid, x.uname, count(*)
from Users x, Picture y
where x.uid = y.author  and y.size > 1000000 and x.city = 'Denver'
group by x.uid, x.uname;
```

v. **No**

Your answer:

Users(uid, uname, city)	Artwork(wid, title, pdf)
Picture(pid, author, size, pdf)	Author(aid,wid,aname)

(f) The company merges with a local adertising company in Denver, which has a database of artistic pictures. Each artistic picture has a title, the pdf image, and may have several authors; all authors are from Denver.

```
drop table if exists Artwork;
drop table if exists Author;

create table Artwork (
   wid int primary key,
   title text,
   pdf text);

create table Author (
   aid int primary key,
   wid int not null references Artwork(wid),
   aname text not null);
```

i. (10 points) Some users and pictures occur in both databases. Write a SQL query that finds all common user,picture pairs. Two users are considered to be the same if their names and cities are the same; two pictures are considered to be the same if their pdf's are the same. Your query should return `uname, uid, aid, pid, wid`: the user name (which is the same in both databases), its two keys in `Users` and `Author`, and the two keys of the identical picture in `Picture` and `Artwork`.

Solution:

```
insert into Artwork values(101, 'Artwork by Alice and Fred', 'abcd');
insert into Artwork values(102, 'Artwork by Fred',           'xyzu');

insert into Author values(201, 101, 'Alice');
insert into Author values(202, 101, 'Fred');
insert into Author values(203, 102, 'Fred');

select distinct x.uid, v.aid, x.uname, y.pid, v.wid
from Users x, Picture y,
     Artwork u, Author v
where x.uid = y.author and u.wid = v.wid and x.city = 'Denver'
  and x.uname = v.aname and y.pdf = u.pdf;
```

2 points off for missing to test `x.city = 'Denver'`
3 points off for missing some join conditions
1-2 points off for unnecessary (but correct) subqueries
Many students answered correctly.

Users(uid, uname, city)	Artwork(wid, title, pdf)
Picture(pid, author, size, pdf)	Author(aid,wid,aname)

ii. (10 points) The new company creates a new schema for the integrated data:

```
create table NewUsers (
    nuid int primary key,
    nuName text not null,
    city text not null);

create table NewPicture (
     npid int primary key,
     title text,
     size int,
     pdf text);

create table Authored (
     nuid int not null references newUsers(nuid),
     npid int not null references NewPicture(npid));
```

Write a sequence of SQL queries that inserts all the data from the two old databases into the new database. You do not need to eliminate duplicates: that is, you will insert all the records from `Users`, `Picture` into the new schema, then will insert all the records from `Artwork`, `Author` into the new schema, without worrying about duplicates. All keys in the `Users`, `Picture` database are distinct from the keys in the `Artwork`, `Author` database.

Solution:

```
insert into NewUsers(nuid, nuName, city)
     select uid as nuid, uname as nuName, city from Users;

insert into NewPicture(npid, title, size, pdf)
     select pid as npid, null as title, size, pdf from Picture;

insert into Authored(nuid, npid)
     select author as nuid, pid as npid from Picture;

insert into NewUsers(nuid, nuName, city)
     select aid as nuid, aname as nuName, 'Denver' as city from Author;

insert into NewPicture(npid, title, size, pdf)
     select wid as npid, title, null as size, pdf from Artwork;

insert into Authored(nuid, npid)
     select aid as nuid, wid as npid from Author;
```

Most students got the high level idea, but few got the details right.

3 to 5 points off for major SQL mistakes.
1-3 points off for unnecessary joins
1 point off for missing `'Denver'`

(This page is intentionally left blank.)

2 Database design

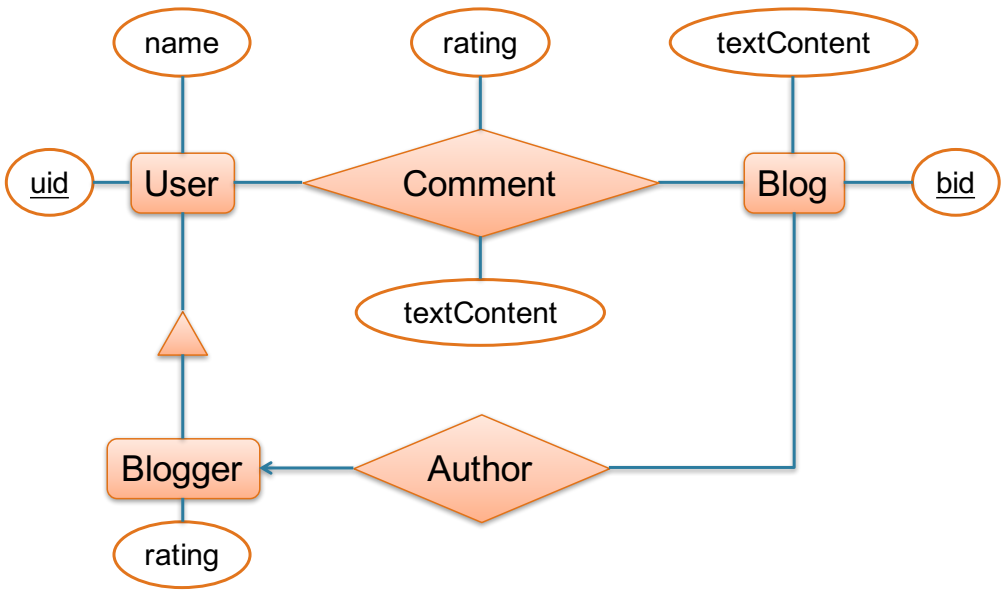
2. (30 points)

   (a) (5 points) You are running a startup company allowing users to post blogs, and to comment on other users' blogs. Design the E/R diagram for your company's data. Your database should store information about users, bloggers, and blogs, and encode the following information:

   - Every user has a `user ID` and a `name`.
   - Every blog has a `blog ID`, a `text content` and one `author`, who is a blogger.
   - Every blogger is a user.
   - Every blogger also has a `rating` attribute.
   - Users may comment on blogs.
   - Every comment has an optional `text content`, and/or an optional `rating`.

   Design the E/R diagram for this database.

Solution:



(b) (10 points) Consider the following table:

FirstName	FirstInitial	LastName	LastInitial	FullName	FullAddress	ZipCode
"Alice"	"A"	"Levy"	"L"	"Alice Levy"	"45h St, Seattle, 98185",	"98185"
"Bob"	"B"	"Levy"	"L"	"Bob Levy"	"45h St, Seattle, 98185",	"98185"
"Alice"	"A"	"Davis"	"D"	"Alice Davis"	"Oak Ave, Seattle, 98185",	"98185"
"Carol"	"C"	"Davis"	"D"	"Carol Davis"	"Sunnyvale, 94085",	"94085"
"Carol"	"C"	"Louis"	"L"	"Carol Louis"	"Oak Ave, Seattle, 98185",	"98185"

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others.

Solution:

```
FirstName -> FirstInitial
LastName -> LastInitial
FirstName, LastName -> FullName
FullAddress -> ZipCode
FullName -> FirstName, LastName, FullAddress
```

(c) (10 points) Using the functional dependencies you have identified at the previous question, decompose the relation in BCNF. Show your work, and the final normalized schema including all keys. Then represent the relation instance below in your normalized schema:

FirstName	FirstInitial	LastName	LastInitial	FullName	FullAddress	ZipCode
"Alice"	"A"	"Levy"	"L"	"Alice Levy"	"45h St, Seattle, 98185",	"98195"
"Bob"	"B"	"Levy"	"L"	"Bob Levy"	"45h St, Seattle, 98185",	"98195"
"Alice"	"A"	"Davis"	"D"	"Alice Davis"	"Oak Ave, Seattle, 98185",	"98195"
"Carol"	"C"	"Davis"	"D"	"Carol Davis"	"Sunnyvale, 94085",	"94085"
"Carol"	"C"	"Louis"	"L"	"Carol Louis"	"Oak Ave, Seattle, 98185",	"98195"

Solution: The BCNF decomposition is:

R1(FirstName,FirstInitial)
R2(LastName,LastInitial)
R3(FullName,FirstName,LastName)
R4(FullName,FullAddress)
R5(FullAddress,ZipCode)

```
drop table if exists R;
create table R (
  FirstName text,
  FirstInitial text,
  LastName text,
  LastInitial text,
  FullName text,
  FullAddress text,
  ZipCode int
);

insert into R values('Alice', 'A', 'Levy', 'L', 'Alice Levy', '45h St, Seattle, 98185', 98195);
insert into R values('Bob', 'B', 'Levy', 'L', 'Bob Levy', '45h St, Seattle, 98185', 98195);
insert into R values('Alice', 'A', 'Davis', 'D', 'Alice Davis', 'Oak Ave, Seattle, 98185', 98195);
insert into R values('Carol', 'C', 'Davis', 'D', 'Carol Davis', 'Sunnyvale, 94085', 94085);
insert into R values('Carol', 'C', 'Louis', 'L', 'Carol Louis', 'Oak Ave, Seattle, 98185', 98195);

drop table if exists R1;
drop table if exists R2;
drop table if exists R3;
drop table if exists R4;
create table R1(FirstName text primary key, FirstInitial text);
create table R2(LastName text primary key, LastInitial text);
create table R3(FullName text primary key, FirstName text, LastName text);
create table R4(FullName text primary key, FullAddress text);
create table R5(FullAddress text primary key, ZipCode int);

insert into R1 (select distinct FirstName, FirstInitial from R);
insert into R2 (select distinct LastName, LastInitial from R);
insert into R3 (select distinct FullName, FirstName, LastName from R);
insert into R4 (select distinct FullName, FullAddress from R);
insert into R5 (select distinct FullAddress, ZipCode from R);

select * from R1;
--  firstname | firstinitial
-- -----------+--------------
--  Carol     | C
--  Alice     | A
--  Bob       | B

select * from R2;
```

```
--  lastname | lastinitial
-- ----------+------------
--  Louis    | L
--  Davis    | D
--  Levy     | L

select * from R3;
--   fullname   | firstname | lastname
-- -------------+-----------+----------
--  Bob Levy    | Bob       | Levy
--  Alice Levy  | Alice     | Levy
--  Alice Davis | Alice     | Davis
--  Carol Louis | Carol     | Louis
--  Carol Davis | Carol     | Davis

select * from R4;
--   fullname   |       fulladdress
-- -------------+-------------------------
--  Bob Levy    | 45h St, Seattle, 98185
--  Alice Davis | Oak Ave, Seattle, 98185
--  Carol Davis | Sunnyvale, 94085
--  Alice Levy  | 45h St, Seattle, 98185
--  Carol Louis | Oak Ave, Seattle, 98185

select * from R5
--        fulladdress       | zipcode
-- -------------------------+---------
--  Sunnyvale, 94085        |   94085
--  45h St, Seattle, 98185  |   98195
--  Oak Ave, Seattle, 98185 |   98195


-- Final check that the decomposition is lossless:
select * from R1 natural join R2 natural join R3 natural join R4 natural join R5;
-- OK
```

(d) (5 points) Consider the following database schema:

```
create table customer (
    cid int primary key,
    cname text,
    city text);
create table product (
    pid int primary key,
    pname text,
    price int);
create table orders (
    cid int references customer,
    pid int references product);
```

Consider the following query:

```
select x.cid, x.cname, x.city, z.pid, z.pname, z.price
from customer x, orders y, product z
where x.city = 'Seattle'
  and x.cid = y.cid
  and y.pid = z.pid
  and z.price = 100;
```

Write all functional dependencies that hold on the query answer.

Solution:

```
cid -> cname
pid -> pname
cname -> city
pname -> city
city -> price
price -> city
```

To use in SQL:

```
drop table if exists orders;
drop table if exists customer;
drop table if exists product;
```

3 Semistructured Data and JSon

3. (10 points)

(a) For each small data instance below indicate whether we should call it relational data, or semi-structured data:

i. (1 point) Is this data:

```
---------------------------
| name  | email    | age  |
---------------------------
| Alice | a@li.ce  | 31   |
| Bob   | b@ob.com | 24   |
| Carol | NULL     | 37   |
| David | da@vid.el| NULL |
| Erol  | e@rol    | 22   |
---------------------------
```

relational or semi-structured? i. **Relational**

ii. (1 point) Is this data:

```
name,email,age
Alice,a@li.ce,31
Bob,b@ob.com,24
Carol,,37
David,da@vid.el,
Erol,e@rol,22
```

relational or semi-structured? ii. **Relational**

iii. (1 point) Is this data:

```
      {"Person"
         [{"name":"Alice","email":"a@li.ce","age":31},
          {"name":"Bob","email":"b@ob.com","age":24},
          {"name":"Carol","age":37},
          {"name":"David","email":"da@vid.el"},
          {"name":"Erol","email":"e@rol","age":22}
         ]
      }
```

iii. **Relational**

relational or semi-structured?

iv. (1 point) Is this data:

```
{"Person"
   [{"name":"Alice","email":["a@li.ce","alice@gmail.com"],"age":31},
    {"name":"Bob","email":"b@ob.com","age":24},
    {"name":"Carol","age":37},
    {"name":{"first":"David","last":"Brown"}, "email":"da@vid.el"},
    {"name":"Erol","email":"e@rol","age":22}
   ]
}
```

iv. **Semi-structured**

Relational or semi-structured?

(b) Consider the following applications. For each, indicate whether you would use a relational data model, or a semistructured data model:

i. (3 points) You work for a large, national financial institution. Every night, all databases from the local branches of your institution sent over the internet all their updates for the day to a central server. The updates are complex in structure and may involve accounts, users, financial transactions, etc. What data model would you use to model the data exchange?

i. **Semi-structured**

Semi-structured or relational?

ii. (3 points) You work for a large online shopping company, and maintain their Orders database. They have tens of millions of orders, and need to access about 2-300 orders per second; orders are retrieved either by the order ID, or by the date, or by the customer. What data model would you use for the orders database?

ii. **Relational**

Semi-structured or relational?

4 Transactions

4. (50 points)

(a) For each statement below, indicate whether it is true or false:

i. (3 points) In a static database, every serializable schedule is also conflict-serializable.

Answer Yes/No: i. **No**

ii. (3 points) In a static database, every conflict-serializable schedule is also serializable.

Answer Yes/No: ii. **Yes**

iii. (3 points) SQL Lite uses optimistic concurrency control.

Answer Yes/No: iii. **No**

iv. (3 points) In a static database, strict Two-Phase-Locking is guaranteed to produce a serializable schedule.

Answer Yes/No: iv. **Yes**

v. (3 points) In a static database, strict Two-Phase-Locking is guaranteed to produce a conflict-serializable schedule.

Answer Yes/No: v. **Yes**

vi. (3 points) In a static database, strict Two-Phase-Locking is guaranteed to avoid deadlocks.

Answer Yes/No: vi. **No**

(b) For each of the schedules below, indicate whether they are conflict-serializable. If you answer *yes*, then give the equivalent serial order of the transactions. Show your work.

i. (6 points) Is this schedule conflict-serializable? Show your work; if you answer 'yes', then indicate a serialization order.

```
R1(A), R1(B), W1(A), R2(B), W2(D), R3(C), R3(B), R3(D), W2(B), W1(C), W3(D)
```

Solution: No: there is the cycle $3 \xrightarrow{B} 2 \xrightarrow{D} 3$ in the dependency graph.

ii. (6 points) Is this schedule conflict-serializable? Show your work; if you answer 'yes', then indicate a serialization order.

```
R1(A), R1(B), W1(A), R2(B), W2(A), R3(C), R3(B), R3(D), W2(B), W1(C), W3(D)
```

Solution: Yes: the dependency graph is acyclic and a serialization order is $T3, T1, T2$.

(c) A scheduler uses the strict two-phase locking protocol. In each of the cases below, indicate whether the scheduler may result in a deadlock. If you answer *yes*, then give an example of a schedule that results in deadlock.

i. (5 points) Can these transactions result in deadlock?

T1:	W1(A), W1(C), CO1
T2:	W2(B), W2(D), CO2
T3:	W3(A), W3(B), CO3
T4:	W4(D), W4(A), CO4

Answer 'yes' or 'no'. If you answer 'yes' then also indicate a schedule that results in deadlock:

Solution:
`W2(B),W3(A),W4(D),W2(D) -- now T2, T3, T4 are deadlocked.`
Transaction T1 is not needed.

ii. (5 points) Can these transactions result in deadlock?

T1:	W1(A), W1(C), CO1
T2:	W2(B), W2(D), CO2
T3:	W3(A), W3(B), CO3
T4:	W4(B), W4(C), CO4
T5:	W5(C), W5(D), CO5

Answer 'yes' or 'no'. If you answer 'yes' then also indicate a schedule that results in deadlock:

Solution: No: all transactions acquire the elements in the order `A,B,C,D` and therefore they avoid deadlock.

(d) (10 points) We run the four transactions below concurrently on sqlite. Consider the following schedule:

Time	T1	T2	T3	T4
1	begin transaction			
2	select * from R where A = 1			
3		begin transaction		
4		select * from R where A = 2		
5	update R set B=10 where A=1			
6			begin transaction	
7			select * from R where A = 3	
8	commit			
9				begin transaction
10				select * from R where A = 4
11		commit		
12			commit	
13				commit

Circle the first action that will not be permitted by sqlite. Then modify the schedule, by filling out the table below, to reflect the actual schedule on sqlite. You can only delay actions, not move them earlier in time.

Time	T1	T2	T3	T4
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

Solution:

T4 is not allowed to begin until T1 commits:

Time	T1	T2	T3	T4
1	begin transaction			
2	select * from R where A = 1			
3		begin transaction		
4		select * from R where A = 2		
5	update R set B=10 where A=1			
6			begin transaction	
7			select * from R where A = 3	
8				
9				
10				
11		commit		
12			commit	
13	commit			
14				begin transaction
15				select * from R where A = 4
16				commit

5 Parallel Data Processing

5. (50 points)

(a) We are running a MapReduce job over HDFS with a block size of 100KB. The input file has 1TB= 10^{12} Bytes. Answer each of the questions below.

i. (3 points) How many map tasks will the MapReduce system create by default? If there is no default, then indicate so.

i. 10^7

Number of map tasks:

ii. (3 points) How many reduce tasks will the MapReduce system create by default? If there is no default, then indicate so.

ii. **No default**

Number of reduce tasks:

iii. We are running a MapReduce job over HDFS with a block size of 100KB. Most of the computation time is spend in the Map phase; the Reduce phase is very fast. Answer each question below:

α) (4 points) Our input file has size 1TB= 10^{12} Bytes. When we run the MapReduce job on 10 workers, the job takes 500 minutes. How long will the job take if we use 100 workers?

α) **50 minutes**

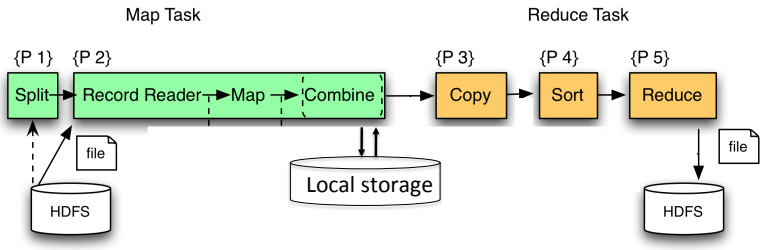
Write the number of minutes:

β) (4 points) Our input file has size 50KB= $5 \cdot 10^4$ Bytes. When we run the MapReduce job on 10 workers, the job takes 10 minutes. How long will the job take if we use 100 workers?

β) **10 minutes**

Write the number of minutes:

For the next few questions, recall the steps of a MapReduce job from the lecture notes:



iv. (3 points) The *Copy* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish. (Recall that MapReduce implements the shuffle phase by having each reducer copy its values from the mappers.)

iv. **Yes**

Yes or no?

v. (3 points) The *Sort* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish.

v. **No**

Yes or no?

vi. (3 points) The *Reduce* phase of the reduce tasks may start immediately after the first map tasks finish without having to wait for all map tasks to finish.

vi. **No**

Yes or no?

(b) The following questions compare MapReduce to Spark. For each statement indicate whether it is true or false.

i. (3 points) In order to cope with worker failure, MapReduce stores all intermediate results to disk.

i. **True**

True or False?

ii. (3 points) In order to cope with worker failure, Spark stores all intermediate results to disk.

ii. **False**

True or False?

iii. (3 points) If a worker fails during the execution of a MapReduce program, then the entire program needs to be restarted.

iii. **False**

True or False?

iv. (3 points) If a worker fails during the execution of a Spark program, then the entire program needs to be restarted.

iv. **False**

True or False?

v. (3 points) Which of the following three Spark transformations corresponds to the `map` phase of a MapReduce program?

1. $\texttt{map}(f) : \texttt{RDD}[T] \Rightarrow \texttt{RDD}[U]$, where $f : T \Rightarrow U$.
2. $\texttt{flatMap}(f) : \texttt{RDD}[T] \Rightarrow \texttt{RDD}[U]$, where $f : T \Rightarrow \texttt{Seq}[U]$.
3. $\texttt{map}(f) : \texttt{RDD}[T] \Rightarrow \texttt{RDD}[T]$, where $f : T \Rightarrow \texttt{Bool}$.

v. **2**

Answer 1,2, or 3:

vi. What is the difference between $\texttt{RDD}[T]$ and $\texttt{Seq}[T]$? Select all answers that apply.

α) (3 points) An RDD can be longer than a Sequence.

α) **False**

True or false?

β) (3 points) An RDD is distributed while a Sequence is stored on a single node.

β) **True**

True or false?

γ) (3 points) An RDD can be nested, e.g. we can have a data type $\texttt{RDD}[(K, \texttt{RDD}[T])]$.

γ) **False**

True or false?

δ) (3 points) A Sequence can be nested, e.g. we can have a data type $\texttt{Seq}[(K, \texttt{Seq}[T])]$.

δ) **False**

True or false?