FINAL: QUESTION 1

Problems 1 through 3 are an exercise in running mongod's, replica sets, and an exercise in testing of replica set *rollbacks*, which can occur when a former primary rejoins a set after it has previously had a failure.

Get the files from Download Handout link, and extract them. Use a.bat instead of a.sh on Windows.

Start a 3 member replica set (with default options for each member, all are peers). (a.sh will start the mongod's for you if you like.)

```
$ # if on unix:
$ chmod +x a.sh
$ ./a.sh
```

You will need to initiate the replica set next.

Run:

```
$ mongo --shell --port 27003 a.js
```

```
> // ourinit() will initiate the set for us.
> // to see its source code type without the parentheses:
> ourinit
>
> // now execute it:
> ourinit()
```

We will now do a test of replica set rollbacks. This is the case where data never reaches a majority of the set. We'll test a couple scenarios.

Now, let's do some inserts. Run:

```
> db.foo.insert( { _id : 1}, {writeConcern : { w : 2 } } )
> db.foo.insert( { _id : 2}, {writeConcern : { w : 2 } } )
> db.foo.insert( { _id : 3}, {writeConcern : { w : 2 } } )
```

Note: if 27003 is not primary, make it primary -- using `rs.stepDown()` on the mongod on port 27001 (perhaps also `rs.freeze()`) for example.

Next, let's shut down that server on port 27001:

```
> var a = connect("localhost:27001/admin");
> a.shutdownServer()
> rs.status()
```

At this point the mongod on 27001 is shut down. We now have only our 27003 mongod, and our arbiter on 27002, running.

Let's insert some more documents:

```
> db.foo.insert( { _id : 4 } )
> db.foo.insert( { _id : 5 } )
> db.foo.insert( { _id : 6 } )
```

Now, let's restart the mongod that is shut down. If you like you can cut and paste the relevant mongod invocation from `a.sh`.

Now run `ps` again and verify three are up:

```
$ ps -A | grep mongod
```

Now, we want to see if any data that we attempted to insert isn't there. Go into the shell to any member of the set. Use `rs.status()` to check state. Be sure the member is "caught up" to the latest optime (if it's a secondary). Also on a secondary you might need to invoke `rs.slaveOk()` before doing a query.)

Now run:

```
> db.foo.find()
```

to see what data is there after the set recovered from the outage. How many documents do you have?

1

SUBMIT