1. **Response time experiment (seq_request.sh)**

Average End to End Response Times, no caching			
	Buy	Lookup	Search
Item 1	0.036129	0.013081	
Item 2	0.033040	0.013377	
Item 3	0.034209	0.013217	
Item 4	0.034005	0.012911	
Topic 1			0.012885
Topic 2			0.012939

Average End to End Response Times, caching			
	Buy	Lookup	Search
Item 1	0.031504	0.003144	
Item 2	0.027580	0.002981	
Item 5	0.032389	0.002964	
Item 7	0.039875	0.002961	
Topic 1			0.002904
Topic 2			0.002866

The results from Lab 2 of 1000 sequential request was copied here for comparison. Caching improves the end-to-end response time on lookup and search, decreasing from about .012 to .0029, for approximately a 4x speedup. There was hardly any change in buy operations, which makes sense since caching cannot be used there.

2. **Caching experiment (cache_timing.py)**

	Writes without invalidation	Writes causing invalidation	Overhead
Time(s)	0.00432	0.00386	-0.00046

Writes without invalidation were timed by issuing 100 consecutive writes, while writes with invalidation were timed by issuing 100 writes that were each preceded by reads. We note that there isn't any positive overhead - this is because in our implementation, cache invalidation just deletes the invalid cache information. This effect is negligible.

	Read with cache	Read with cache miss	Overhead
Time(s)	0.00342	0.00363	0.00021

Reads with the cache were timed by issuing consecutive reads, while reads with cache misses were timed by issuing reads preceded by writes. There is a small overhead from cache miss.

3. **Fault tolerance experiment**

First we start the system without crashes, and issue some buy operations:

```
bought book 'RPCs for Dummies'

bought book 'RPCs for Dummies'

bought book 'How to get a good grade in 677 in 20 minutes a
day'

etc...
```

Next we crash the order server replica on port 6001:

```
Disabling primary order replica on server 1 and querying other
server
6001/tcp:            83466
Spider mode enabled. Check if remote file exists.
--2019-04-24 22:15:36--  http://128.119.243.164:6001/orders
Connecting to 128.119.243.164:6001... failed: Connection
refused.
```

We now issue more buy requests, similar to before. The database of order server 0 is out of sync, since it cannot be accessed. The other replica masks this fault since buys continue to succeed

```
Connecting to 128.119.243.164:6001... failed: Connection
refused.
bought book 'Xen and the Art of Surviving Graduate School'

bought book 'RPCs for Dummies'

etc...
```

After issuing these buys, we restart and recover order server 0. It gives the following message to indicate that it is syncing with order server 1:

```
Order_0 sync up with replica1
```

Finally, we check that resynchronization has occured by using the check API call to each order server, although manual inspection indicates that order server 0 database is synced:

```
wget -qO- http://128.119.243.147:6001/check
"Database synced with peer!"
```

We conduct the same experiment, except with a crashed catalog server. We repeat the same process intially - start the system without any crashes, issue some buys, then crash a catalog server:

```
Disabling primary catalog replica on server 1 and querying
other server
6006/tcp:            83470
Spider mode enabled. Check if remote file exists.
--2019-04-24 22:29:34--  http://128.119.243.147:6006/query/1
Connecting to 128.119.243.147:6006... failed: Connection
refused.
```

While the catalog replica is down, we issue both lookup requests and buys, since both types of requests would need to be routed to the functioning replica server. Finally, we check recovery and resynchronization by checking itemcounts:

```
wget -qO-http://128.119.243.147:6006/query/1
wget -qO-http://128.119.243.164:6002/query/1
wget -qO-http://128.119.243.147:6006/query/2
wget -qO-http://128.119.243.164:6002/query/2
etc...
```