1) Install Memcached
2) Run Memcached (on Mac: \texttt{/opt/homebrew/opt/memcached/bin/memcached -l localhost}) \texttt{\textasciitilde open a new terminal window once this is running in background}
3) Ensure Memcached is running (on Unix: \texttt{ps -ef | grep \textasciitilde i memc})
4) Connect to Memcached through telnet (\texttt{telnet localhost 11211}) \texttt{\textasciitilde open a new terminal window once this is running in background}
5) Run \texttt{stats} to retrieve statistics of our Memcached server (e.g., uptime, view how many times certain queries were run, etc...)
6) Let's add a key-val pair (\texttt{<command> <key> <flag> <exp_time> <byte_size> <no_op>})
   - First (add key): \texttt{set foo 0 3600 3}
   - Second: Press enter to actually enter the value on the next line
   - Third (add val): \texttt{bar}
7) Memcached will return \texttt{STORED}
8) Run \texttt{stats} again to analyze differences
9) Run get foo and see return
10) Run delete foo and wait for Memcached to let you know it’s gone
11) Run get foo and nothing is returned
12) Run stats to see that get_misses is now 1
13) Let's add another key-val (note: different between add and set is that add will not change value of a key but set will):
    - add num 0 3600 2 0
    - 50
14) Let's get num
15) Now let's append:
    - append num 0 3600 2
    - 25
16) What do you think get num will return? (Ans: 5025)
17) Now let's prepend:
    - prepend num 0 3600 2
    - 44
18) What do you think get num will return? (Ans: 445025)
19) Now let's replace:
    - replace num 0 3600 2
    - 40
20) What do you think get num will return? (Ans: 40)
21) We can increment num by 2: \texttt{incr num 2}
22) And then decrement by 2 as well: \texttt{decr num 2}
23) Let's clear our cache: \texttt{flush\_all}
24) Running stats again shows that while curr\_items are 0, total\_items are still 5
25) Let's close our connection with memcached through telnet: \texttt{quit}
26) Let's open it back up with command from 4) and notice that there are still total\_items == 5
27) Let's kill our current Memcached server by canceling the process we spawned in step 2 in another terminal window
28) Now we can restart our server by simply running the command from step 2 and
29) reinitiating a connection with telnet - at this point, total_items should be 0
30) The great thing about memcached is that it works with pretty much all languages and there are myriad interfaces... Let's experiment with one in Python!
31) Make sure a Memcached server is running and a telnet connection is open in another window
32) Open a new window and create a venv and specify a python version (e.g., conda create --name test python=3.5)
33) Install the python-memcached interface into your venv: conda install -c anaconda python-memcached
34) Enter a Python shell: python
35) Write the following script to import, initiate a client, and set/get key/val pair:

```python
>>> import memcache
>>> mc = memcache.Client(['127.0.0.1:11211'], debug=0)
>>> mc.set('greet','Hello World')
True
>>> mc.get('greet')
'Hello World'
```
36) Double-check existence of 'greet' by running 'get greet' in your telnet terminal window
37) This shows Memcached is language agnostic in real-time - great!
38) Let's now leverage our interface language by assigning Memcached to values we can extend...
39) Let's create an object with multiple keys: mc.set_multi({'name':'John Doe', 'email':'jdoe@cornell.edu'})
40) We can:

```python
>>> mc.get('name')
'John Doe'
```

```python
>>> mc.get_multi(['name','email'])
{'name': 'John Doe', 'email': 'jdoe@cornell.edu'}
```
41) Let's assign this key/val in our network cache to a Python value: person = mc.get_multi(['name', 'email'])
42) We can:

```python
>>> person.keys()
dict_keys(['name', 'email'])
```

```python
>>> person.values()
dict_values(['John Doe', 'jdoe@cornell.edu'])
```
43) More:
mc.delete_multi(['name'])
1
mc.get('name')
mc.get('email')
jdoe@cornell.edu
mc.flush_all()
mc.get('email')