1. Sometimes, it's necessary to impose restrictions on network usage. This could include simple filters that refuse connections, electronic notifications to a sysadmin if a prohibited resource were accessed, or, as in the case of this task, something sillier. Sometimes it's to try to enforce productivity, sometimes to censor content, and sometimes to discourage specific behaviours that might fall between the two. If someone is wasting time on, say, Wikipedia (we'll say only the English version, for the sake of simplicity), then they should perhaps be nudged in the direction of responsibility.

For this task, your job is to write what will appear to be a simple DNS server, with the following requirements and exceptions:

- The current three websites one shouldn't visit while trying to remain productive are kingdomofloathing.com (69.16.150.211), en.wikipedia.org (208.80.154.225), and tvtropes.org (216.151.212.103); all of which will henceforth be known as prohibited
- One could potentially decide on additional “bad” sites later on, so ensure that your solution is coded to readily allow for additional entries (i.e. hardcoding the addresses is fine, but do it in an array of Strings or something)
- It will only use UDP (i.e. you don't need to bother with the TCP version. If a message would be truncated, that's fine)
- It will actually defer all of the “work” to a real DNS server (use 8.8.8.8)
- It must be able to accept an indefinite number of requests (i.e. you can't have it simply process one request and then quit. Put it in a loop!)
- For all requests, simply pass them through to the “real” DNS server
  - For all non-prohibited requests, it must pass the DNS server's response back to the requesting client intact
  - For any prohibited request, it must modify the DNS server's response to replace all offending addresses with 139.57.100.6
- You may assume only one question per request, but must account for multiple answers
- Your filter must only affect actual IP addresses; MX, CNAMEs, etc. must remain unaltered
- Java is recommended. You can use another compiled language, but then you may find testing to be a greater nuisance
- Though it's fun, you may wish to avoid testing your program out with your computer/browser. If you forget what you changed, it could be annoying to restore. Instead, just use dig (e.g. on sandcastle. Refer to the end for a sample execution)
- For a minor refresher on coding for UDP, refer to the 3P01 slides
- For tips on how DNS works, refer to one (or both) of the following: http://www.tech-juice.org/2011/06/22/the-dns-protocol-explained/ http://www.netfor2.com/dns.htm

2. Even if one were to configure one's router/gateway to use a box running software similar to that in #1 as its DNS server, how could one easily circumvent it? Explain.

3. Tell me about WAP. What was it for? What were its limitations? What is WML? Don't just recite Wikipedia pages. Seriously.
Submission:
Both physical and electronic submission are required. The due date applies to both. If you miss either by the posted date/time, expect to receive a zero.
To submit electronically, upload your files to sandcastle, ssh/PuTTY in, navigate to the appropriate folder, and run `submit4p14`.
For your physical submission, include printouts of both code and sample executions, and remember to attach a departmental coverpage before putting it into the dropbox.

Sample Execution for DNS:
```bash
$ dig @127.0.0.1 -p 5754 +ignore en.wikipedia.org
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53812
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
en.wikipedia.org. IN A

;; ANSWER SECTION:
wikipedia-lb.eqiad.wikimedia.org. 3115 IN A 139.57.100.6

;; Query time: 43 msec
;; SERVER: 127.0.0.1#5754(127.0.0.1)
;; WHEN: Tue Jan 29 04:05:55 2013
;; MSG SIZE  rcvd: 120
```

Note that the canonical names were still passed through to the client.
Also note that I chose 5754 for my port number. Do not even attempt to use 53 on sandcastle!
For more info on how `dig` works, try typing `man dig`. 