# CS107 Assignment 6: Frequently Asked Questions

1. **Question:** You provide us with a lot of code. But what's thread safe and what isn't?

   **Answer:** Excellent question! The url, urlconnection, and streamtokenizer functions aren't thread safe, but this shouldn't be a problem. The url and urlconnection functions are nothing more than constructors and destructors, and there's no reason for two threads to be racing to construct and dispose of shared urls and urlconnections. streamtokenizer functions aren't thread-safe, but you don't need to (and I'm sure you weren't inclined to) share streamtokenizers with multiple threads so that they'd be racing with competing nextToken calls. All the functions in html-utils.h are thread-safe. The vector and hashset are not thread-safe, and because all of your threads will be competing to access and update several hashsets and vectors, you need to use Semaphores to prevent race conditions.

2. **Question:** How many files are allowed to be open at the same time? Do I need to worry about this? **OR** I'm getting response code 0's all the time. What gives?

   **Answer:** By default, each process is limited to 128 open FILE *s. Three of these are used for stdin, stdout, and stderr, so the most that rss-news-search cane have open at any one time is 125. However, you can bump that limit up to the max by typing

   ```
   limit descriptors 256
   ```

   at the command prompt. The data files have been updated so that rss-feeds-tiny.txt is small enough so that a test run won't max out on the number of open files.

   The ultimate solution to this has you implement that portion of the assignment that limits the number of active HTTP connections to 36. This is a little trickier than you might think, because you have to protect yourself against deadlock if a good number of calls to ParseArticle call themselves recursively via an HTTP response of 30x. If you get this working, and you get it working well, then your rss-news-search application is all set to digest even the large rss-feeds.txt file.

3. **Question:** Are we allowed to use global variables?

   There's no compelling reason to use globals in this particular program, so no!

   The sequential solution we've provided packages all of the data structures into a single **rssDatabase** struct, and the address of that struct is passed down from **main**, through the XML parser as user data, so that it's available to the callback functions and all the code called by them. You can add any **Semaphore**s to the **rssDatabase** structure so they, like the **hashset**s and **vector**s they contain, are available to all threads.

4. **Question:** Sometimes my gdb just quits and says "suspended (tty input)". How do I fix it?

   GDB's got a strange bug. Just type "fg" (foreground) and it should start running again.

5. **5. Question:** How can we use purify?

   We've created a new Makefile that **sort of** works with purify. Go to your assignment directory and grab it from the assignment directory with the command

   ```
   cp /usr/class/cs107/assignments/assn-6-rss-news-search/Makefile .
   ```

   Run **`make pure`** as usual to compile, and then execute **`rss-news-search.purify`**.

There are some bugs and quirks with Purify and threads, so we can't guarantee it will work, but you can give it a try if you're running into segfaults and other problems. Let us know if you have problems running it and we'll see if we can help out.

In particular, note that purify will report some false positives. You may see things such as

1. IPW (Invalid Pointer Write) in some library function
2. leaked memory in the thread and semaphore packages even though you clean-up all your semaphores
3. array bounds violations in IsWordWellFormed