1. pthread_t is an integer type that is the id of a thread. It is unique within the current process. They can be reused after a terminated thread has been joined, or a detached thread has terminated. http://man7.org/linux/man-pages/man3/pthread_self.3.html
2. A thread safe function is one that functions correctly when accessed by multiple threads, avoiding race conditions, lockout, and starvation. Thread safe functions also protect shared data. Example:
   Non-thread-safe:
   ```
   int x = 0;

   void increment() {
       x++;
   }
   ```

   Thread-safe:
   ```
   #include <pthread.h>
   int x = 0;
   pthread_mutex_t mutex;

   void increment() {
       pthread_mutex_lock(&mutex);
       x++;
       pthread_mutex_unlock(&mutex);
   }
   ```
   http://pubs.opengroup.org/onlinepubs/009695399/functions/pthread_mutex_lock.html
3. ```
   Void increment(struct foo *) {
       pthread_mutex_lock(&(foo->f_lock));
       foo->f_count++;
       pthread_mutex_unlock(&(foo->f_lock));
   }

   struct foo * foo_create() {
       struct foo* str;
       foo->f_count = 0;
       pthread_mutex_init(&(str->f_lock));
       return str;
   }
   ```
4. GCD C blocks seem very similar to C++11 lambdas. They're both functions that can be stored into variables and passed into other functions as such. C++11 lambdas require you to declare which local variables they have access to while GCD C blocks have access to all local variables by default.
   ```