Using Helgrind

1. Create a simple multithreaded program that increments a counter. Identify data races using Helgrind.

Locks

2. Use pthread mutex to write a fully correct working version of a multi-threaded program that shares a variable count across multiple threads. Increment the variable in each thread for a given number of times in a loop.

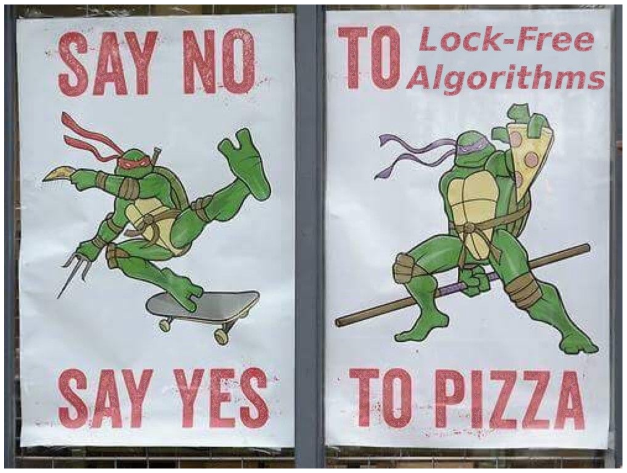
Lock-free Programming: A tip of the iceberg

3. Use atomic types in c11 to create the correct concurrent thread program to increment the counter.
   Reference:
   https://lumian2015.github.io/lockFreeProgramming/c11-features-in-currency.html

   FAQ :

   - "Why use locks when I can use atomic types?"
     - The above example is equivalent to the "Hello World" program in the lock-free programming world. It becomes way more complex as you go deeper. Although it gives tremendous performance boosts, it suffers from the ABA problem. To know more read "Fear and Loathing in Lock-Free Programming" and "ABA problem"

   As a general mantra:



Using assembly in C!

4. Write a simple single-threaded program in C to increment a number by 1. The increment should be done via inline assembly.
5. Use “cmpxchgl” instruction in assembly to build your own lock. Test for its correctness by comparing to Q3

Refer to: Using Inline Assembly in C/C++