- What are the main differences between threads and processes?

Threads and processes are very similar, however, they share one key difference which changes their potential dramatically, while a new process doesn't share content with its parent, a thread shares data. So while multiple threads and processes can run the same code, threads will share the same globals, static local variables, and pointers to an object that are passed into the thread's run function, the only thing that isn't shared is local variables. But since this is the case and the OS decides what to schedule in an inscrutable manner, it makes us question how to write code so that the threads don't change the other's data.

- What does a thread do when it tries to `join` but the other thread hasn't completed yet?

Join is the "wait for the other thread to complete" operation. If the main thread creates a new thread then main performs a join, it waits for the new thread to complete. While execution on the thread performing the join is blocked on the join, as in it will sleep until the other thread exits.

- What's the difference between concurrency and parallelism?

Concurrency refers to a program that can always produce correct results even when multiple threads are running in an unspecified order. While parallelism refers to just the process of multiple threads or processes running at the same time. This is basically the difference in reliability of a program to do what is desired even when other threads or processes are running together, and the idea that the processes can run together in general.

- Global variables are shared among threads, but local variables are not. Describe why this is the case.

Each thread has its own stack. This is convenient so as when multiple threads are running the same function they don't interfere with the other threads' locals.