**HW1 Pt1**

Goroutine vs Thread:

A Goroutine is a function or method that executes on its own and simultaneously with any other goroutines present in a go program, while a thread is a subset of a process; it is a single, independent path of execution and it can run simultaneously with other threads in the same process. Advantages of threads are that it is available in basically every programming language and operating system, they have control which CPU core it is run on, it has lots of operating system tooling and debuggers, and it has direct hardware mapping as an OS scheduler has the information on all the threads. Some disadvantages for threads are that they are larger in memory usage compared to goroutines, they can’t be ran as much at a time compared to goroutines, they are prone to deadlocks, and blocking a thread is “expensive” as it can stop a full OS thread from running. Advantages for goroutines are that compared to threads they use much less memory and they are dynamically allocated, many more goroutines can be run at a time compared to threads, they are faster to switch as they are managed by go’s runtime as opposed to the operating system, they have more simple concurrency compared to threads as they communicate using their own channels which reduces shared memory complexity, and they are easier to use compared to threads as their creation syntax is incredibly simple. Disadvantages of goroutines are that they have less control than threads (they have no direct way to control priority, affinity, or scheduling), they are limited to go, they require go runtime and a go garbage collector to be ran which adds overhead compared to using threads, and they are more complicated to debug as many more can be ran at a time, which may make debugging difficult.

3 Sources:

<https://www.geeksforgeeks.org/go-language/golang-goroutine-vs-thread/>

Geeksforgeeks is credible because it is an educational website dedicated to teaching topics in computer science and engineering. Their content is created by mentors from academic organizations and institutions.

<https://jayconrod.com/posts/128/goroutines-the-concurrency-model-we-wanted-all-along>

This source is credible as it is written by a software engineer who works at EngFlow. He also used to work on creating the Go language, so he knows what he’s talking about when talking about pros and cons of goroutines.

<https://medium.com/@sairavitejachintakrindi/goroutines-and-threads-exploring-concurrency-in-go-370d609038c>

This source is credible as it is an article that is written by Sai Ravi Teja, who I found out is an applied scientist at Boston Dynamics with a masters degree in electrical and computer engineering.