HW Group 4

HW3

Part 1: Go Design Principle

“Don't communicate by sharing memory; share memory by communicating.” Basically, it means that instead of different threads all accessing and changing the same memory data while coordinating with locks or other memory synchronization methods, communication should be the primary method via channels. So basically, memory is shared by using either pointers, values, or handles over channels. This reduces the necessity of using locks and reduces the risks of data traces. With this “method”, usually only one goroutine oversees a piece of memory at a time using the communication channel.

Advantages of this design principle are safety, clearer ownership, and decoupled concurrency. This method is safer because the program is less likely to have multiple goroutines simultaneously changing the same object which reduces the chance of unsynchronized concurrent access. This method has clearer ownership because a channel is more “explicit” on what owns the data at a given time when carrying a pointer. Decoupled concurrency is an advantage because goroutines, unlike threads, don’t need knowledge of locks or memory layout, and instead they just respond to messages, which can make concurrency easier to achieve.

Disadvantages of this design principle are message passing, pointer leak, and complexity. Sending messages, allocating data, or pointer passing can have larger memory costs than different approaches. Pointers can leak because if a pointer is sent from a channel but is still kept as a reference in the sender, it can reintroduce shared memory which can cause issues. This design principle is more complex in certain algorithms, which can cause it to be harder to develop and get working properly, especially if an inexperienced developer is trying to utilize this design principle. Also, this design principle brings deadlock risks, as if channels are misused, deadlocks could occur.

3 Sources and why they’re credible:

<https://go.dev/talks/2012/splash.article>

This article is credible because it is written by google themselves and it is an official document. It also has that quote “Don’t communicate by sharing memory, share memory by communicating.” In the article.

<https://tip.golang.org/doc/codewalk/sharemem/>

This is a “codewalk” which walks you through go code while also explaining approaches and such. It is also written officially by google and go developers, so it is definitely credible.

<https://research.swtch.com/gomm>

This article is written by Russ Cox, who was one of the go language designers. Because of this, he is credible on talking about almost anything go related.