[Race conditions](http://en.wikipedia.org/wiki/Race_condition" \t "_blank) are among the most insidious and elusive programming errors. They typically cause erratic and mysterious failures, often long after the code has been deployed to production. While Go's concurrency mechanisms make it easy to write clean concurrent code, they don't prevent race conditions. Care, diligence, and testing are required. And tools can help.

Race conditions occur when different processes try to access the same piece of data in memory. This can lead to non-deterministic outcomes depending on how the processes are scheduled.

Outcome of x in this program depends on non-deterministic ordering. If you run this program you will see that the value of x will change depending on how the go scheduler decides to order their execution. A race condition occurs in this case because both go routines are accessing the same piece of data, x in this case. Depending on how the scheduler decides to

execute these two routines the final outcome of x may be 0, 1 or some other value.