Homework 03

ECE 473/573, Fall 2024

Due Date: 10/06 (Sun.) by the end of the day (Chicago time)

Consider the following Go code where two goroutines `ping` and `pong` are supposed to output "ping" and "pong" alternatively by collaborating via channels.

```
package main

import (
    "fmt"
)

func ping(in <-chan bool, out chan<- int, n int) {
    for i := 0; i < n; i++ {
        <-in // wait for signal from pong or start
        fmt.Printf("ping %d\n", i)
        out <- i // let pong do its job
    }
    close(out) // notify pong of done
}

func pong(in <-chan int, out chan<- bool, done chan<- struct{}) {
    for i := range in { // get i from ping
        fmt.Printf("pong %d\n", i)
        out <- false // let ping do its job
    }
    close(done) // notify main of done
}

func main() {
    pi := make(chan bool) // line B
    po := make(chan int)
    done := make(chan struct{})
    defer close(pi)

    go ping(pi, po, 10)
```

```
    go pong(po, pi, done)

    fmt.Println("Start!")
    // line A

    <-done
}
```

1. *(1 point)* The desired output should be

   ```
   Start!
   ping 0
   pong 0
   ping 1
   pong 1
   ping 2
   pong 2
   ping 3
   pong 3
   ping 4
   pong 4
   ping 5
   pong 5
   ping 6
   pong 6
   ping 7
   pong 7
   ping 8
   pong 8
   ping 9
   pong 9
   ```

   However, go reports a deadlock after displaying "Start!". Add a line at line A so that `ping` and `pong` will start to output messages as desired. (Hint: send something to `ping`!)

2. *(1 point)* Still, go reports a deadlock after dispalying all desired messages. Modify line B to resolve the issue. Explain why. (Hint: what if you send something to a channel but no one is receiving?)