// Answer for exercise 1 of Race Conditions.

package main

import (

"fmt"

"math/rand"

"sync"

"time"

)

// numbers maintains a set of random numbers.

var numbers []int

// mutex will help protect the slice.

var mutex sync.Mutex

// init is called prior to main.

func init() {

rand.Seed(time.Now().UnixNano())

}

// main is the entry point for the application.

func main() {

// Number of goroutines to use.

const grs = 3

// wg is used to manage concurrency.

var wg sync.WaitGroup

wg.Add(grs)

// Create three goroutines to generate random numbers.

for i := 0; i < grs; i++ {

go func() {

random(10)

wg.Done()

}()

}

// Wait for all the goroutines to finish.

wg.Wait()

// Display the set of random numbers.

for i, number := range numbers {

fmt.Println(i, number)

}

}

// random generates random numbers and stores them into a slice.

func random(amount int) {

// Generate as many random numbers as specified.

for i := 0; i < amount; i++ {

n := rand.Intn(100)

// Protect this append to keep access safe.

mutex.Lock()

{

numbers = append(numbers, n)

}

mutex.Unlock()

}

}