// Create a program that declares two anonymous functions. One that counts down from

// 100 to 0 and one that counts up from 0 to 100. Display each number with an

// unique identifier for each goroutine. Then create goroutines from these functions

// and don't let main return until the goroutines complete.

//

// Run the program in parallel.

package main

import (

"fmt"

"runtime"

"sync"

)

func init() {

// Allocate one logical processor for the scheduler to use.

runtime.GOMAXPROCS(1)

}

func main() {

// Declare a wait group and set the count to two.

var wg sync.WaitGroup

wg.Add(2)

fmt.Println("Start Goroutines")

// Declare an anonymous function and create a goroutine.

go func() {

// Count down from 100 to 0.

for count := 100; count >= 0; count-- {

fmt.Printf("[A:%d]\n", count)

}

// Tell main we are done.

wg.Done()

}()

// Declare an anonymous function and create a goroutine.

go func() {

// Count up from 0 to 100.

for count := 0; count <= 100; count++ {

fmt.Printf("[B:%d]\n", count)

}

// Tell main we are done.

wg.Done()

}()

// Wait for the goroutines to finish.

fmt.Println("Waiting To Finish")

wg.Wait()

// Display "Terminating Program".

fmt.Println("\nTerminating Program")

}