**Lab Practive -IV(13-03-2025)**

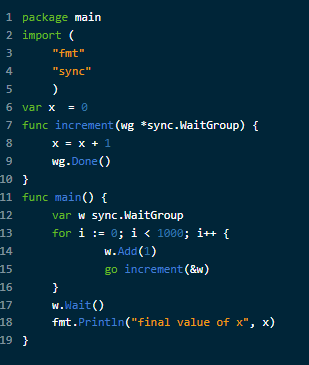
**Go Channels and Mutex**

1. Structure the Go program for the number "456" such that the sum of squares of each digit are calculated in a separate Goroutine, sum of cubes of each digit in another Goroutine and the final summation of squared-sum and cube-sum happens in the main Goroutine. Use Channels for communicating the input from the main Goroutine and outputs to the main Goroutine.

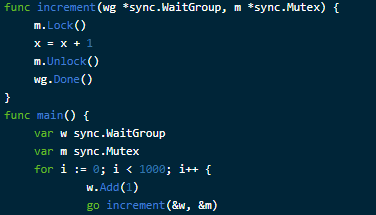
2. Write a Go program that calculates the sum of an array using multiple goroutines. The array should be split into N parts, with each part being summed by a separate goroutine. Use channels to collect the partial sums and compute the final sum in the main function.

3. Implement a parallel matrix multiplication using goroutines, channels and with proper synchronization. Given two matrices A (size MxN) and B (size NxP), compute the product matrix C = A × B using multiple goroutines. Each goroutine computes the value of **one element** in the result matrix C. Use a channel to store the computed values before merging them into the final matrix.

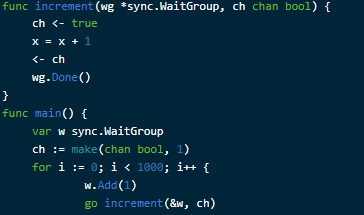
4. a. Execute the below code and check for race Condition..



b. Solution 1: with mutex



c. Solution 2: using Channels



5. Create a concurrent bank account system where multiple goroutines can deposit and withdraw money from a shared account. Use a mutex to prevent race conditions. Implement a BankAccount struct with a balance and a sync.Mutex. Create functions for Deposit(amount int) and Withdraw(amount int), ensuring that they are thread-safe. Start multiple goroutines performing deposits and withdrawals simultaneously. Print the final balance after all transactions.