# Matching.go

**Vad händer om man tar bort go-kommandot från Seek-anropet i main-funktionen?**

Teori: Det kommer att fungera som förut bortsett från att seek metoden kommer att

köras en gång itaget för varje name, med tanke på att det inte görs i separata gorutioner

Vad som händer: Det fungerar som det ska, men tror att det tar längre tid.

Anledning: Bara för att det inte är go före seek() så måste det köras en gång i taget vilket gör att det tar mycket längre tid för att köra allt.

**Vad händer om man byter deklarationen wg := new(sync.WaitGroup) mot var wg sync.WaitGroup och parametern wg \*sync.WaitGroup mot wg sync.WaitGroup?**

Teori: Det borde inte fungera för nu sätter du in hela wg som parameter och inte dess pekare.

Vad som händer: Du får ett deadlock efter ett tag, dem första saker skrivs ut, men sista ge deadlock

Anledning: Det är för att wg.Add(len(people)) innehåller 5 och när alla go routiner körs så får alla

routiner själva variabeln wg som är 5 och alla minskar det med 1, vilket gör att routine 1 minskar det till 4 och routin 2 minskar det

från 5 till 4 och så gör det alla. Det resulterar att wg går bara ner till 4 och sen händer inget, vilket ger deadlock.

**Vad händer om man tar bort bufferten på kanalen match?**

Teori: Då kommer det troligen att få deadlock.

Vad som händer: Du får ett deadlock efter sista personen i people

Anledning: Detta händer för att Eva skickas in i en kanal,

men nu finns det inget som hämtar ur Eva, vilket gör att det kommer att stanna där för evigt

och sedan ge deadlock

**Vad händer om man tar bort default-fallet från case-satsen i main-funktionen?**

Teori: Inget kommer att hände för inget finns i default

Vad som händer: Programmet fungerar som det ska. Men när man har ett jämnt antal personer så får man deadlock

Anledning: Detta är för att när man kommer till select satsen så finns det inget att skicka ut från kanalen, vilket gör att man fastnar i select satsen och får deadlock.

# Julia.go

Programmet skriver ut hur många CPUer jag har och hur snabbt programmet är.

När min version körs så tar det ca 10 sec och när originalet körs så tar det ca 28 sec. Detta säger att mitt program är 18 sec snabbare.

Min dator har 4 logiska CPUer.

# Client.go

## Funktionen Read innehåller två lömska buggar. Beskriv i detalj vad som är fel.

En bugg här är att när programmet körs och det blir en timeout,

då kommer res att skrivas som res = &Response{"Gateway timeout\n", 504}.

Men när Read() är klar så körs go routinen fortfarande och kan sedan ge ett värde till res, vilket inte ska ske när metoden är klar.

Den andra buggen hittade jag inte, och jag letade säkert med tre andra i typ 5 timmar strek.

## Skriv en korrekt Read-funktion. Glöm inte dokumentationskommentaren.

Lösningen för första buggen finns dokumenterat i koden.

## Implementera funktionen Multiread.

Kolla koden.