Telkom University	**Tugas 1** **Pemrograman Jaringan (CSH4V3)** **----------** **Semester Ganjil 2019 - 2020** **Dosen: Aulia Arif Wardana, S.Kom., M.T. (UIW)**

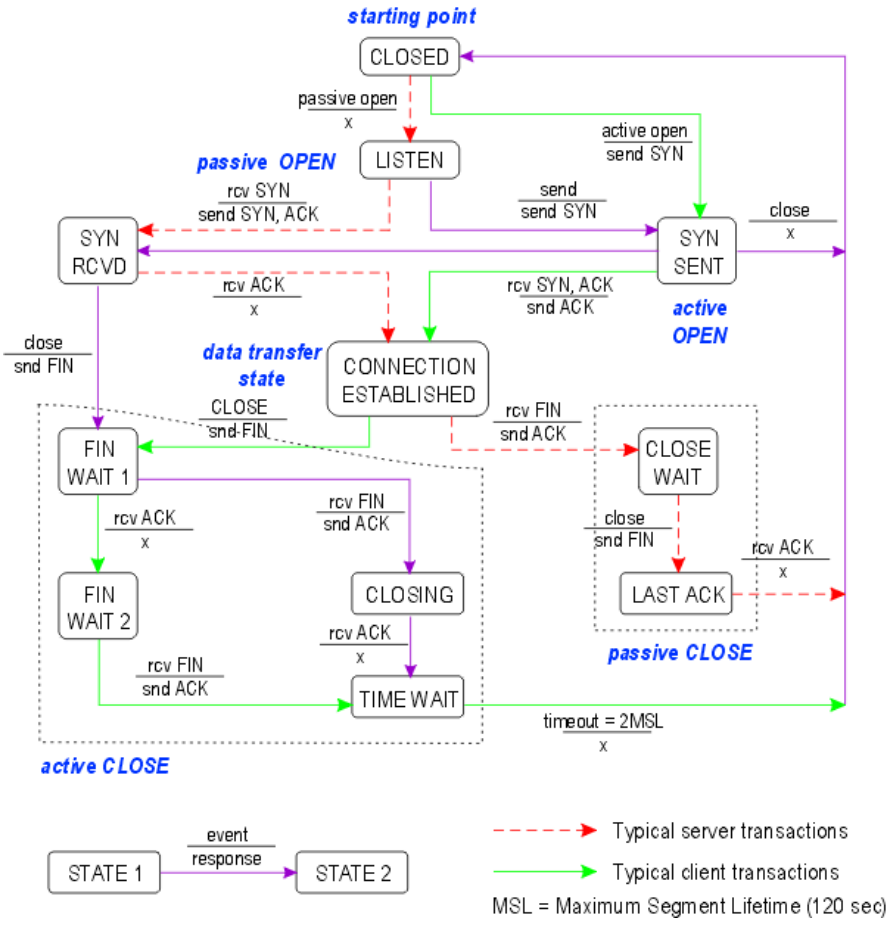
Berdo'alah sebelum mengerjakan. Dilarang berbuat curang.
Tugas ini untuk mengukur kemampuan anda, jadi kerjakan dengan sepenuh hati.
Selamat belajar, semoga sukses !

Nama Mahasiswa:	NIM:	Nilai:
Lukman Budiman	**1301164725**	
……	……	……

Siapkan tools berikut sebelum mengerjakan:

1. Go Programming Language (https://golang.org/dl/).
2. Visual Studio Code (https://code.visualstudio.com/) atau LiteIDE (https://github.com/visualfc/liteide).
3. Harus menggunakan linux dengan distro fedora (https://getfedora.org/id/workstation/).
4. Buatlah git repository pada https://github.com/ kemudian push semua kode dan hasil laporan anda ke dalam repository github yang sudah anda buat.
5. Kumpulkan link repository github tersebut sebagai tanda bahwa anda mengerjakan tugas modul ini.
6. Link repository harus berbeda untuk setiap tugasnya. Buatlah markdown yang rapi disetiap repository tugas yang anda kumpulkan.
7. Printscreen program harus dari desktop anda sendiri, dan harus dari linux yang sudah diinstall. Jika tidak, maka harus mengulang pengerjaan tugasnya.
8. Jangan lupa untuk menuliskan NAMA dan NIM pada laporan.
9. Laporan berbentuk PDF dan dikumpulkan pada link repository github beserta kodenya.

Soal No 1



15. Jelaskan maksud diagram finite state machine dari TCP Connection diatas!

Jawaban: Diagram diatas menjelaskan tentang koneksi berlangsung melalui serangkaian states selama masa pakainya. States yang dipakai adalah : LISTEN, SYN-SENT, SYNRECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, dan state fiksi CLOSED. CLOSED disebut fiksi karena mewakili keadaan ketika tidak ada TCP, dan oleh karena itu, tidak ada koneksi. Secara singkat arti dari state adalah :

LISTEN	Mewakili menunggu permintaan koneksi dari

	TCP dan port jarak jauh
SYN-SENT	Mewakili menunggu permintaan koneksi yang cocok setelah mengirim permintaan koneksi.
SYN-RECEIVED	Mewakili menunggu konfirmasi permintaan koneksi konfirmasi setelah keduanya menerima dan mengirim permintaan koneksi.
ESTABILISHED	Mewakili koneksi terbuka, data yang diterima dapat dikirim ke pengguna. Keadaan normal untuk fase transfer data dari koneksi
FIN-WAIT-1	Mewakili menunggu permintaan penghentian koneksi dari TCP jarak jauh, atau pengakuan atas permintaan pemutusan koneksi yang sebelumnya dikirim.
FIN-WAIT-2	Mewakili, menunggu permintaan penghentian koneksi dari TCP jarak jauh.
CLOSE-WAIT	Mewakili, menunggu permintaan penghentian koneksi dari user lokal.
CLOSING	Mewakili menunggu konfirmasi permintaan penghentian koneksi dari jarak jauh TCP.
LAST-ACK	Mewakili menunggu pemberitahuan permintaan penghentian koneksi sebelumnya dikirim ke TCP jarak jauh (yang mencakup pengakuan penghentian koneksinya permintaan).

TIME-WAIT	Mewakili menunggu waktu yang cukup untuk memastikan TCP remote menerima pengakuan atas permintaan penghentian koneksi.
CLOSED	Tidak mewakili kondisi koneksi sama sekali.

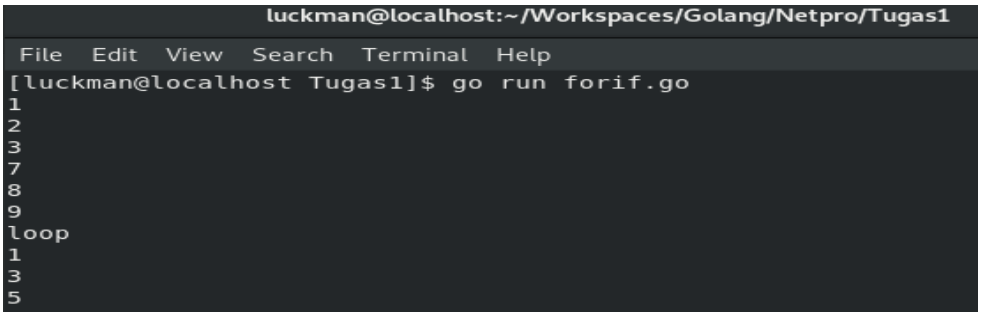
Koneksi TCP berkembang dari satu kondisi ke kondisi lain sebagai respons terhadap berbagai peristiwa. Peristiwa adalah panggilan pengguna, OPEN, SEND, RECEIVE, CLOSE, ABORT, dan STATUS; segmen yang masuk, terutama yang mengandung bendera SYN, ACK, RST dan FIN; dan batas waktu.

Soal No 2 (for dan if/else)

```go
package main

import "fmt"

func main() {

    i := 1
    for i <= 3 {
        fmt.Println(i)
        i = i + 1
    }

    for j := 7; j <= 9; j++ {
        fmt.Println(j)
    }

    for {
        fmt.Println("loop")
        break
    }

    for n := 0; n <= 5; n++ {
        if n%2 == 0 {
            continue
        }
        fmt.Println(n)
    }
}
```

```go
package main

import "fmt"

func main() {

    if 7%2 == 0 {
        fmt.Println("7 is even")
    } else {
        fmt.Println("7 is odd")
    }

    if 8%4 == 0 {
        fmt.Println("8 is divisible by 4")
    }

    if num := 9; num < 0 {
        fmt.Println(num, "is negative")
    } else if num < 10 {
        fmt.Println(num, "has 1 digit")
    } else {
        fmt.Println(num, "has multiple digits")
    }
}
```

Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

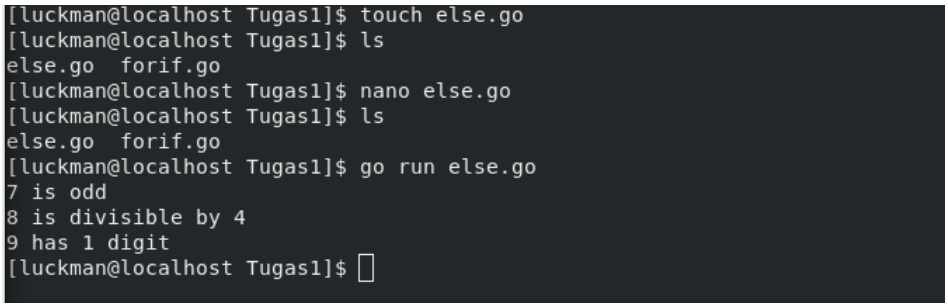


Jawaban:

Untuk program for dan if menjelaskan tentang perulangan. Dimana pada for yang pertama dimulai dari I = 1, lalu program melakukan loop selama I <= 3, maka yang dihasilkan adalah bilangan 1,2,3

Lalu untuk for yang kedua adalah j = 7, selama j<=9 maka bilangan yang dihasilkan adalah 7,8,9

Terakhir yaitu for yang ketiga adalah n = 0, selama n<=5 tetapi ada perkondisian dimana n jika di modulo 2 sama dengan 0, sehingga bilangan yang dihasilkan adalah bilangan ganjil yaitu 1,3,5



Untuk program yang else terdapat perkondisisan pada if else yang pertama yaitu apabila 7%2==0 maka 7 is even, tetapi karena 7%2 masih mempunyai sisa maka hasilnya adalah “7 is odd”

Lalu untuk if yang kedua terdapat kondisi 8%4 ==0, karena 8%4 tidak menyisakan sisa, maka hasilnya adalah 8 is divisible by 4

Terakhir kondisi yang ketiga dimana num = 9. Jika num < 0 maka dia akan print “9 is negative”, jika num < 10 maka “9 has 1 digit” , jika tidak keduanya maka “bilangan has multiple digits”

Soal No 3 (array dan function)

27.
```go
package main

import "fmt"

func main() {

    var a [5]int
    fmt.Println("emp:", a)



    a[4] = 100
    fmt.Println("set:", a)
    fmt.Println("get:", a[4])

    fmt.Println("len:", len(a))

    b := [5]int{1, 2, 3, 4, 5}
    fmt.Println("dcl:", b)

    var twoD [2][3]int
    for i := 0; i < 2; i++ {
        for j := 0; j < 3; j++ {
            twoD[i][j] = i + j
        }
    }
    fmt.Println("2d: ", twoD)
}
```

```go
package main

import "fmt"

func plus(a int, b int) int {


    return a + b
}

func plusPlus(a, b, c int) int {
    return a + b + c
}



func main() {

    res := plus(1, 2)
    fmt.Println("1+2 =", res)

    res = plusPlus(1, 2, 3)
    fmt.Println("1+2+3 =", res)
}
```

37. Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

```
luckman@localhost:~/Workspaces/Golang/Netpro/Tugas1
File  Edit  View  Search  Terminal  Help
[luckman@localhost Tugas1]$ touch array.go
[luckman@localhost Tugas1]$ nano array.go
[luckman@localhost Tugas1]$ go run array.go
emp: [0 0 0 0 0]
set: [0 0 0 0 100]
get: 100
len: 5
dcl: [1 2 3 4 5]
2d: [[0 1 2] [1 2 3]]
[luckman@localhost Tugas1]$
```

program pertama adalah array dimana array yang pertama dipanggil dengan var a[5] dengan nilai yang kosong berjumlah 5, didapat emp : [0 0 0 0 0]

lalu yang kedua masih mirip-mirip dengan yang pertama tetapi pada yang kedua terdapat perbedaan dimana int a[4] = 100, data yang kosong terdapat 4 dan dilanjutkan dengan angka 100, sehingga hasilnya adalah set : [0 0 0 0 100]

lalu yang ketiga memanggil array a[4] maka hasilnya adalah 100

lalu yang ke empat itu adalah menghitung panjang nilai yang ada di (a) sehingga hasilnya adalah 5

lalu yang ke lima terdapat array b = [5] dengan int{1, 2,3,4,5} lalu dipanggil dan dicetak b maka outputnya adalah array dcl : [1 2 3 4 5]

lalu yang terakhir adalah array dua dimensi dimana terdapat perulangan juga di perulangan yang pertama yaitu I harus kurang dari 2 , lalu perulangan yang kedua j harus kurang dari 3, lalu dari perulangan tadi array tersebut dijumlahkan twoD[i][j] = I +j, sehingga hasilnya adalah [0 1 2] [1 2 3]

```
[luckman@localhost Tugas1]$ touch function.go
[luckman@localhost Tugas1]$ nano function.go
[luckman@localhost Tugas1]$ go run function.go
1+2= 3
1+2+3= 6
[luckman@localhost Tugas1]$
```

program function diatas adalah program penjumlahan yang ada di fungsi plus dan plusPlus

program pertama plus mempunyai int a dan b yang akan dijumlahkan, dan dipanggil difungsi main

dan disimpan dalam variabel res = plus(1,2) sehingga hasilnya adalah 1 + 2 = 3

program yang kedua plusPlus mempunyai int a, b, c yang akan dijumlahkan dan dipanggil di fungsi main dan disimpan dalam variabel res = plus(1,2,3) sehingga hasilnya adalah 1 + 2 + 3 = 6

Soal No 4 (struct dan method)

38.
```go
package main

import "fmt"

type person struct {
    name string
    age  int
}

func main() {

    fmt.Println(person{"Bob", 20})

    fmt.Println(person{name: "Alice", age: 30})

    fmt.Println(person{name: "Fred"})

    fmt.Println(&person{name: "Ann", age: 40})

    s := person{name: "Sean", age: 50}
    fmt.Println(s.name)

    sp := &s
    fmt.Println(sp.age)

    sp.age = 51
    fmt.Println(sp.age)
}
```

```go
package main

import "fmt"

type rect struct {
    width, height int
}

func (r *rect) area() int {
    return r.width * r.height
}

func (r rect) perim() int {
    return 2*r.width + 2*r.height
}

func main() {
    r := rect{width: 10, height: 5}

    fmt.Println("area: ", r.area())
    fmt.Println("perim:", r.perim())

    rp := &r
    fmt.Println("area: ", rp.area())
    fmt.Println("perim:", rp.perim())
}
```

48. Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

```
luckman@localhost:~/Workspaces/Golang/Netpro/Tugas1
File  Edit  View  Search  Terminal  Help
[luckman@localhost Tugas1]$ touch strcut.go
[luckman@localhost Tugas1]$ nano strcut.go
[luckman@localhost Tugas1]$ go run strcut.go
# command-line-arguments
./strcut.go:13:35: syntax error: unexpected ], expecting comma or }
[luckman@localhost Tugas1]$ nano strcut.go
[luckman@localhost Tugas1]$ go run strcut.go
{Bob 20}
{Alice 30}
{fred 0}
&{Ann 40}
Sean
50
51
```

Struct merupakan keyword yang memungkinkan membuat deklarasi yang di dalamnya dapat terdapat banyak variabel, hasil deklarasi ini akan menghasilkan user-defined data type yaitu tipe structure yang berguna untuk mendirikan object. Pada fungsi main terdapat output pertama yaitu {Bob 20} dimana langsung dicetak karena sudah sesuai dengan variabel yang sudah dideklarasikan yaitu name dan age. Lalu pada output kedua lebih spesifik lagi dimana dalam programnya ditambahkan variabel name dan juga age dengan hasil yang sama yaitu {Alice 30}, lalu output yang ketiga adalah {fred 0} ini dikarenakan variabel yang ditulis di program hanya variabel name saja, sehingga sistem akan membaca variabel age sebagai nilai 0, lalu untuk output yang keempat terdapat pointer & dilanjutkan dengan {Ann 40} . Lalu yang terakhir adalah outputnya dipanggil dalam variabel s dikarenakan s= person{name: “senam, age: 50} dan yang dipanggil adalah s.name maka yang keluar adalah Sean. Lalu dipanggil dalam variabel sp = &s dengan sp.age maka yang keluar adalah 50, dan sp.age=51 dipanggil

```
[luckman@localhost Tugas1]$ touch method.go
[luckman@localhost Tugas1]$ nano method.go
[luckman@localhost Tugas1]$ go run method.go
area:  50
perim:  30
area:  50
perim:  30
[luckman@localhost Tugas1]$
```

sp.age maka yang keluar adalah 51.

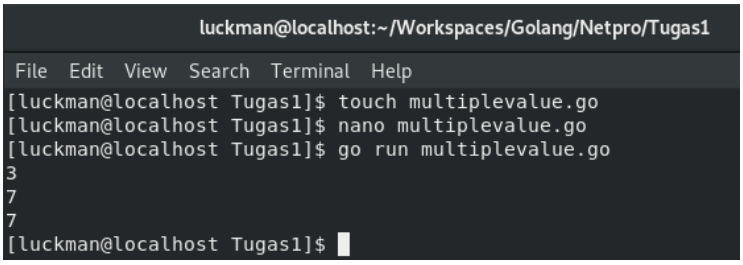
Program method, didalam program terdapat fungsi type rect dengan variabel int width dan height. Lalu terdapat fungsi (r * rect) area() int{ return r.width * r.height},Fungsi ini adalah untuk mengalikan antara width dan height yang nanti akan dipanggil kembali dalam fungsi main. lalu fungsi (r rect) perim() int{ return 2 *r.width + 2*r.height} fungsi ini adalah fungsi perkalian width yang dikali 2 dan ditambah dengan height yang dikali 2 juga, nantinya fungsi ini juga akan dipanggil dalam fungsi main

Soal No 5 (multiple return value dan command line)

```
package main

import "fmt"

func vals() (int, int) {
    return 3, 7
}

func main() {

    a, b := vals()
    fmt.Println(a)
    fmt.Println(b)

    _, c := vals()
    fmt.Println(c)
}
```

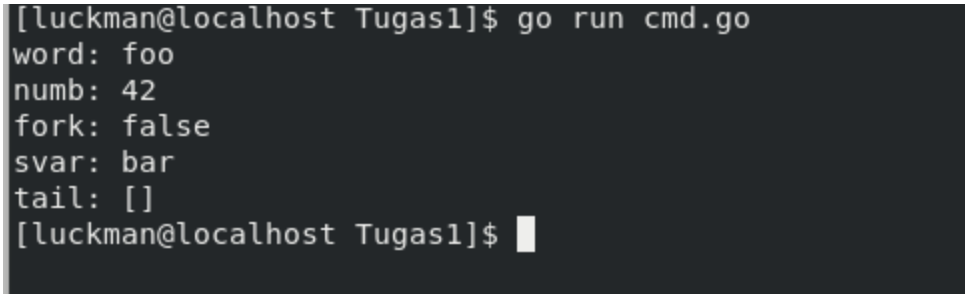
```
package main

import "flag"
import "fmt"


func main() {

    wordPtr := flag.String("word", "foo", "a string")




    numbPtr := flag.Int("numb", 42, "an int")
    boolPtr := flag.Bool("fork", false, "a bool")

    var svar string
    flag.StringVar(&svar, "svar", "bar", "a string var")


    flag.Parse()


    fmt.Println("word:", *wordPtr)
    fmt.Println("numb:", *numbPtr)
    fmt.Println("fork:", *boolPtr)
    fmt.Println("svar:", svar)
    fmt.Println("tail:", flag.Args())
}
```

59. Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!



Jawaban:

pada program multivlereturnvalue diatas dia akan meretrun sebanyak dua kali sesuai parameter fungsi vals, fungsi pertama yaitu a, b:=vals() dengan println(a) dan println(b) maka output keluar adalah 3 , 7 lalu fungsi _, c = vals() dengan println(c) maka hasilnya adalah 7 ini dikarenakan 3 sudah



diganti dengan _,

program diatas adalah hasil dari program command line, program diatas bekerja dengan pengecekan kondisi dimana outputnya adalah default berbeda lagi apabila kita masukan inputan yang berbeda maka hasilnya juga beda

Soal No 6 (simple web application)

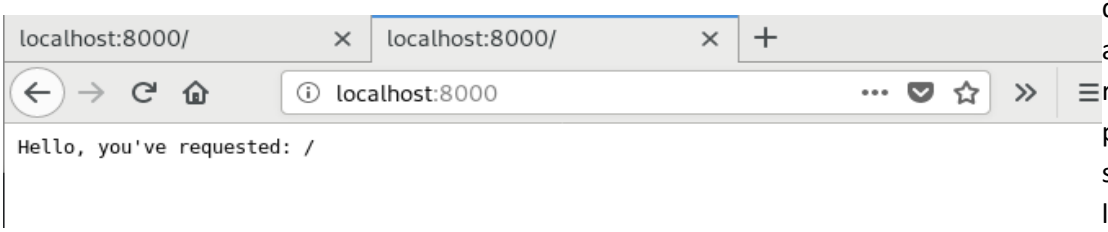
```
package main

import (
        "fmt"
        "net/http"
)

func main() {
        http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
                fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
        })

        http.ListenAndServe(":80", nil)
}
```

67. Sebelum menjalankan program diatas, gantilah port 80 ke port 8000. Buka browser kemudian ketikkan alamat localhost:8000.

68. Jalankan program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

```
[luckman@localhost Tugas1]$ nano simplewebApp.go
[luckman@localhost Tugas1]$ go run simplewebApp.go
```



program diatas akan me-run program sebagai localhost atau menjadi server dengan port 8000, lalu

ini adalah hasil ketika kita mengakses localhost:8000 tadi.

Soal No 7 (create config file)

Buatlah sebuah config file untuk aplikasi web application pada soal no 6 dengan menggunakan library berikut: https://github.com/spf13/viper !

Jelaskan susunan directory dari program serta bagaimana cara untuk melakukan konfigurasi file config yang telah anda buat!

71. Printscreen hasil dan penjelasan kode untuk membuat file config disini!

Jawaban:

Pertama lakukan run “go get -u github.com/spf13/viper” di terminal tanpa tanda kutip

```
[luckman@localhost Tugas1]$ go get -u github.com/spf13/viper
```

Lalu buat file config.json

```
{
    "server":{
        "port": ":8000"
    }
}
```

Lalu lengkapi file yang sudah dibuat tadi pada no. 6 yaitu simplewebApp.go

dengan menambahkan library dari viper.

```go
package main

import (
    "fmt"
    "net/http"

    "github.com/spf13/viper"
)

func main() {
    viper.SetConfigType("json")
    viper.AddConfigPath(".")
    viper.SetConfigName("config")
    viper.ReadInConfig()

    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
    })
    http.ListenAndServe(viper.GetString("server.port"), nil)
}
```

Lalu buka browser dengan mengetikan localhost:8000

maka hasilnya adalah sebagai berikut: