

Tugas 1 Pemrograman Jaringan

Javiar Fasyah – 1301164477

Jawaban Soal No. 1:

Diagram yang dimaksud adalah diagram TCP three-way handshake, yang terjadi saat *client* dan *server* berkomunikasi dalam suatu jaringan.

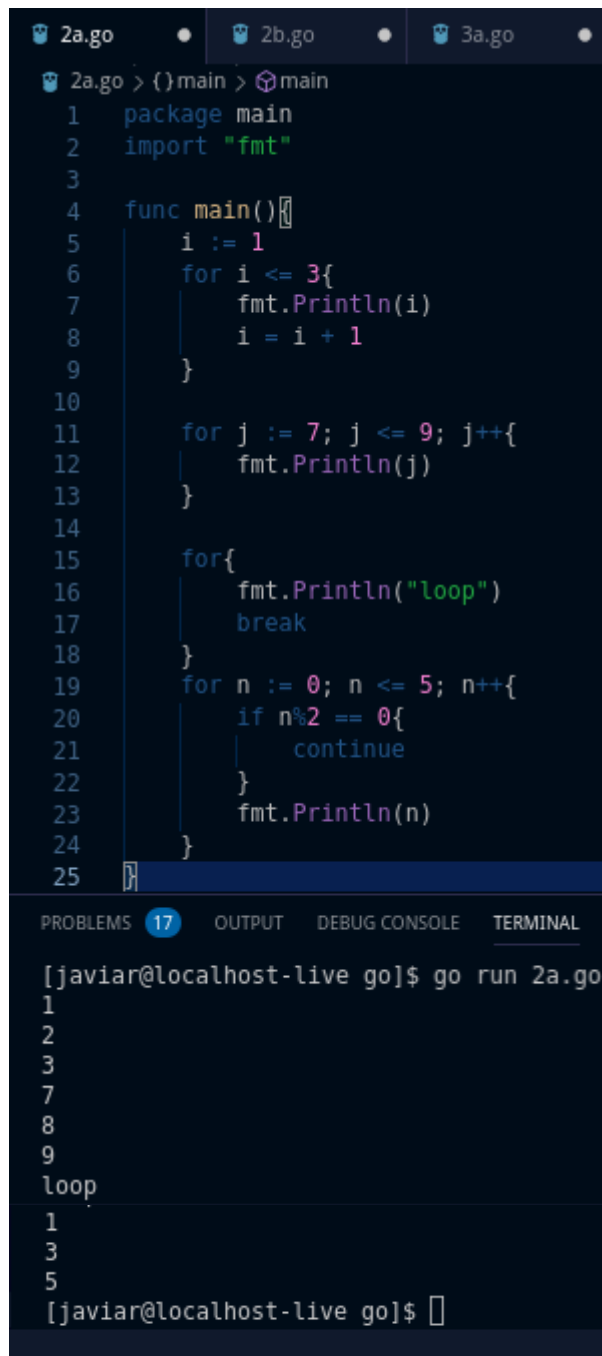
Dari sisi *client*, hal pertama yang dilakukannya setelah aktif adalah mengirim pesan SYN (*synchronize*) kepada *server*. Jika pesan diterima dan *server* menjawab dengan pesan SYN, ACK (*synchronize, acknowledge*), maka kedua entitas tersebut sudah membangun koneksi dan siap berkomunikasi. Disaat *client* menyudahi sesi dengan *server* (mengirim pesan FIN (*finish*) dan *server* menjawab dengan mengirim pesan ACK lalu FIN (atau kebalikannya), maka kedua entitas tersebut resmi terputus koneksinya.

Dari sisi *server*, disaat *server* sudah siap dan menerima pesan SYN dari *client*, maka *server* akan mengirim pesan SYN, ACK untuk memulai koneksi dengan *client*. Setelah mendapat jawaban ACK dari *client*, koneksi diantara keduanya terbangun dan siap berkomunikasi. Disaat *client* ingin menyudahi sesi dan mengirim pesan FIN, *server* akan mengirim pesan ACK lalu mengirim pesan FIN setelah sesi diterminasi. Setelah *client* menjawab ACK, maka koneksi keduanya resmi terputus.

Jawaban Soal No. 2:

Program pertama adalah tentang berbagai macam sintaks iterasi dalam bahasa Go, output yang dihasilkan adalah “1, 2, 3; 7, 8, 9; loop; 1, 3, 5”. Pada iterasi pertama, variabel di-assign nilai diluar sintaks iterasi, iterasi dilakukan hingga batas yang ditentukan dengan mencetak nilai variabel saat iterasi lalu menambah nilai variabelnya hingga memenuhi batas iterasi. Pada iterasi kedua, nilai iterasi di-assign dalam sintaks beserta penambahan nilai variabelnya, iterasi dilakukan hingga batas yang ditentukan sembari mencetak nilai variabel.

Pada iterasi ketiga, terdapat sintaks *break* yang akan menyudahi iterasi sesuai keinginan (dalam kasus ini setelah mencetak string “loop”). Pada iterasi terakhir, iterasi dilakukan mirip dengan pada iterasi kedua, namun dalam pencetakan nilai variabel terdapat sebuah kondisi dimana jika nilai variabel di-mod 2 sama dengan 0 (bilangan genap), maka pencetakan nilai variabel dilewati (dengan sintaks *continue*).



```
2a.go > ({}main > main
1 package main
2 import "fmt"
3
4 func main(){
5     i := 1
6     for i <= 3{
7         fmt.Println(i)
8         i = i + 1
9     }
10
11     for j := 7; j <= 9; j++){
12         fmt.Println(j)
13     }
14
15     for{
16         fmt.Println("loop")
17         break
18     }
19     for n := 0; n <= 5; n++){
20         if n%2 == 0{
21             continue
22         }
23         fmt.Println(n)
24     }
25 }
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 2a.go
1
2
3
7
8
9
loop
1
3
5
[javiar@localhost-live go]$
```

Program kedua adalah tentang penulisan sintaks kondisi pada bahasa Go, output yang dihasilkan adalah “7 is odd; 8 is divisible by 4; 9 has 1 digit”. Pada kondisi pertama, terdapat pengecekan nilai 7 dengan di-mod 2, jika hasilnya 0 maka dicetaklah “7 is even”, jika tidak maka dicetak “7 is odd”. Pada kondisi kedua, terdapat kondisi mirip dengan kondisi pertama (jika 8 di-mod 4 adalah 0, maka dicetak “8 is divisible by 4”), namun jika seandainya 8 di-mod 4 bukan 0, maka program tidak akan mencetak apapun. Pada kondisi terakhir, nilai variabel di-assign didalam sintaks kondisi untuk kemudian dibandingkan, jika variabel kurang dari 0 maka dicetak

“is negative”, jika variabel kurang dari 10 maka dicetak “has 1 digit”, dan jika bukan keduanya maka dicetak “has multiple digits”.



```
2a.go 2b.go 3a.go 3b.go
2b.go > {}main > main
1 package main
2 import "fmt"
3
4 func main(){
5     if 7%2 == 0{
6         fmt.Println("7 is even")
7     } else{
8         fmt.Println("7 is odd")
9     }
10
11     if 8%4 == 0{
12         fmt.Println("8 is divisible by 4")
13     }
14
15     if num := 9; num < 0{
16         fmt.Println(num, "is negative")
17     } else if num < 10{
18         fmt.Println(num, "has 1 digit")
19     } else{
20         fmt.Println(num, "has multiple digits")
21     }
22 }

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL
[javiar@localhost-live go]$ go run 2b.go
7 is odd
8 is divisible by 4
9 has 1 digit
[javiar@localhost-live go]$
```

Jawaban Soal No. 3:

Program pertama adalah tentang array dalam bahasa Go, output yang dihasilkan adalah “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]”. Array pertama di-assign sebagai array integer dan dicetak saat array masing kosong, lalu array tersebut di-assign nilai 100 pada index ke-4 sehingga saat array dicetak kembali terdapat nilai 100 pada index terakhir, nilai pada index terakhir juga dicetak tersendiri dan panjang dari array dicetak dengan sintaks *len*. Array kedua juga merupakan array integer yang di-assign langsung dengan nilai pada masing-masing index, sehingga saat dicetak array menampilkan nilai-nilai dari masing-masing index. Array terakhir merupakan array dua dimensi (2 array dengan masing-masing 3 index), array dua dimensi ini di-assign nilainya dengan iterasi bersarang (iterasi didalam iterasi).

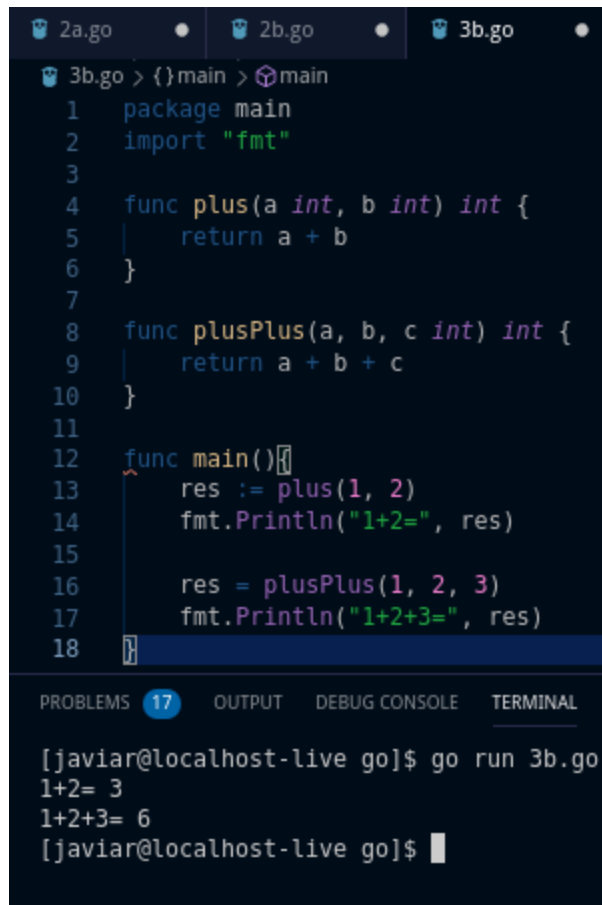


```
3a.go > {}main > main
1 package main
2 import "fmt"
3
4 func main(){
5     var a[5]int
6     fmt.Println("emp:", a)
7
8     a[4] = 100
9     fmt.Println("set:", a)
10    fmt.Println("get:", a[4])
11
12    fmt.Println("len:", len(a))
13
14    b := [5]int{1, 2, 3, 4, 5}
15    fmt.Println("dcl:", b)
16
17    var twoD[2][3]int
18    for i := 0; i < 2; i++){
19        for j := 0; j < 3; j++){
20            twoD[i][j] = i + j
21        }
22    }
23    fmt.Println("2d:", twoD)
24 }
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 3a.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]]
[javiar@localhost-live go]$
```

Program kedua adalah tentang fungsi dalam bahasa Go, outputnya adalah “1+2= 3; 1+2+3= 6”. Fungsi pertama adalah fungsi khusus tipe data integer yang akan mengembalikan hasil penambahan dari dua variabel masukan parameter, parameter pada fungsi ini dideklarasikan tipe datanya satu persatu. Fungsi kedua mirip dengan fungsi pertama yang khusus untuk tipe data integer dan mengembalikan hasil penambahan dari tiga variabel parameter, namun parameter pada fungsi dideklarasikan semuanya pada suatu tipe data.



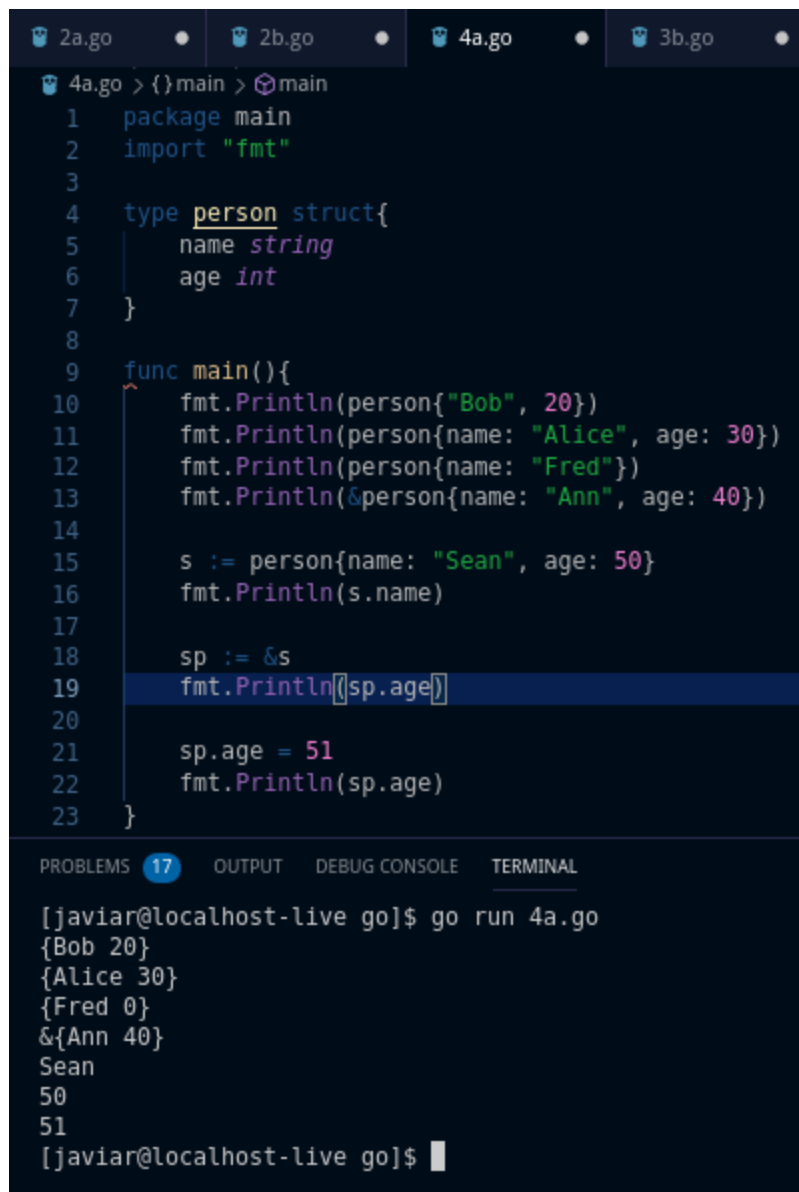
```
3b.go > {}main > main
1 package main
2 import "fmt"
3
4 func plus(a int, b int) int {
5     return a + b
6 }
7
8 func plusPlus(a, b, c int) int {
9     return a + b + c
10 }
11
12 func main() {
13     res := plus(1, 2)
14     fmt.Println("1+2=", res)
15
16     res = plusPlus(1, 2, 3)
17     fmt.Println("1+2+3=", res)
18 }
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 3b.go
1+2= 3
1+2+3= 6
[javiar@localhost-live go]$
```

Jawaban Soal No. 4:

Program pertama adalah tentang tipe data buatan (struct) pada bahasa Go, output yang dihasilkan adalah “{Bob 20}, {Alice 30}, {Fred 0}, &{Ann 40}; Sean, 50, 51”. Tipe data yang dibuat pada program ini adalah tipe data *person* yang terdiri dari atribut string *name* dan integer *age*. Tipe data ini dapat di-assign nilai *name* dan *age*-nya dengan langsung memasukan nilai ke parameter tipe data (contoh: `person{"Bob", 20}`) atau diperjelas dengan atributnya (contoh: `person{name: "Alice", age: 30}`), bisa juga hanya memasukan nilai ke salah satu atributnya (contoh: `person{name: "Fred"}`) sehingga atribut yang lain dimasukan nilai default. Selain pencetakan penuh nilai tipe data, nilai dari atribut bisa dipanggil satu persatu dengan urutan tipe datanya lalu atributnya (contoh: memanggil hanya *name* yakni dengan sintaks `person.name`). pencetakan atribut bisa dilakukan dengan pointer (variabel merujuk ke variabel lain yang tipe datanya *person*) dan dari pointer ini bisa diubah nilai tipe datanya.



```
4a.go > {}main > main
1 package main
2 import "fmt"
3
4 type person struct{
5     name string
6     age int
7 }
8
9 func main(){
10     fmt.Println(person{"Bob", 20})
11     fmt.Println(person{name: "Alice", age: 30})
12     fmt.Println(person{name: "Fred"})
13     fmt.Println(&person{name: "Ann", age: 40})
14
15     s := person{name: "Sean", age: 50}
16     fmt.Println(s.name)
17
18     sp := &s
19     fmt.Println(sp.age)
20
21     sp.age = 51
22     fmt.Println(sp.age)
23 }
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 4a.go
{Bob 20}
{Alice 30}
{Fred 0}
&{Ann 40}
Sean
50
51
[javiar@localhost-live go]$
```

Program kedua adalah tentang method (fungsi yang khusus dimiliki sebuah struct) pada bahasa Go, output yang dihasilkan adalah “area: 50, perim: 30; area: 50, perim: 30”. Method dapat digunakan saat struct sudah diinisialisasi pada suatu variabel dan dipanggil sama dengan cara memanggil atribut dari tipe data (contoh: method *area()* pada tipe data *rect* dipanggil dengan sintaks *rect.area()*). Parameter input yang digunakan method merupakan atribut-atribut yang dimiliki tipe data tersebut. Method juga dapat dipanggil oleh variabel pointer yang merujuk ke variabel dengan tipe data pemilik method tersebut.



```
4b.go > {}main > main
1 package main
2 import "fmt"
3
4 type rect struct{
5     width, height int
6 }
7
8 func (r *rect) area() int{
9     return r.width * r.height
10 }
11
12 func (r rect) perim() int{
13     return 2*r.width + 2*r.height
14 }
15
16 func main(){
17     r := rect{width: 10, height: 5}
18
19     fmt.Println("area:", r.area())
20     fmt.Println("perim:", r.perim())
21
22     rp := &r
23     fmt.Println("area:", rp.area())
24     fmt.Println("perim:", rp.perim())
25 }
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 4b.go
area: 50
perim: 30
area: 50
perim: 30
[javiar@localhost-live go]$
```

Jawaban Soal No. 5:

Program pertama adalah tentang pengembalian nilai lebih dari satu pada bahasa Go, output yang dihasilkan adalah “3, 7; 7”. Pengembalian nilai lebih dari satu ini bisa dilakukan oleh sebuah fungsi, namun cara menerima dari pengembalian nilai lebih dari satu harus dengan cara tuple, dimana terdapat lebih dari satu (sesuai banyak nilai yang dikembalikan fungsi) yang akan menerima nilai dari fungsi tersebut (contoh: fungsi `vals()` memiliki dua nilai yang akan dikembalikan, penerimanya adalah `a, b := vals()`). Nilai yang dikembalikan dari fungsi tersebut, dapat diabaikan salah satunya dengan menggunakan `_` (underscore) pada tuple (contoh: contoh: fungsi `vals()` memiliki dua nilai yang akan dikembalikan namun nilai pertama diabaikan, maka penerimanya adalah `_, b := vals()`).



```
5a.go > {}main > main
1 package main
2 import "fmt"
3
4 func vals() (int, int){
5     return 3, 7
6 }
7
8 func main()
9     a, b := vals()
10    fmt.Println(a)
11    fmt.Println(b)
12
13    _, c := vals()
14    fmt.Println(c)
15
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 5a.go
3
7
7
[javiar@localhost-live go]$
```

Program kedua adalah tentang flag (command line) pada bahasa Go, output yang dihasilkan adalah “word tes, numb 5, bool true, svar h, tail: [b b t]”. Flag berfungsi untuk memproses masukan flag saat menjalankan program. Program akan memproses (mencetak nilai dari flag pada program ini) hanya flag yang di-assign nilainya, jika tidak maka program akan mencetak nilai default dari flag. Flag dapat dideklarasikan dengan di-assign ke variabel atau mendeklarasikan flag lalu merujuk (pointer) ke sebuah variabel. Sintaks yang memproses masukan flag adalah *flag.Parse()*.

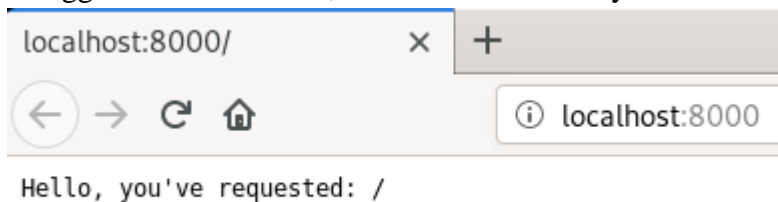

```
5b.go 6.go 7.go
5b.go > {}main > main
1 package main
2 import "flag"
3 import "fmt"
4
5 func main() {
6     wordPtr := flag.String("word", "foo", "a string")
7     numbPtr := flag.Int("numb", 42, "an int")
8     boolPtr := flag.Bool("fork", false, "a bool")
9
10    var svar string
11    flag.StringVar(&svar, "svar", "bar", "a string var")
12    flag.Parse()
13
14    fmt.Println("word", *wordPtr)
15    fmt.Println("numb", *numbPtr)
16    fmt.Println("bool", *boolPtr)
17    fmt.Println("svar", svar)
18    fmt.Println("tail:", flag.Args())
19 }
```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 5b.go -word=tes -numb=5 -fork -svar=h b b t
word tes
numb 5
bool true
svar h
tail: [b b t]
[javiar@localhost-live go]$
```

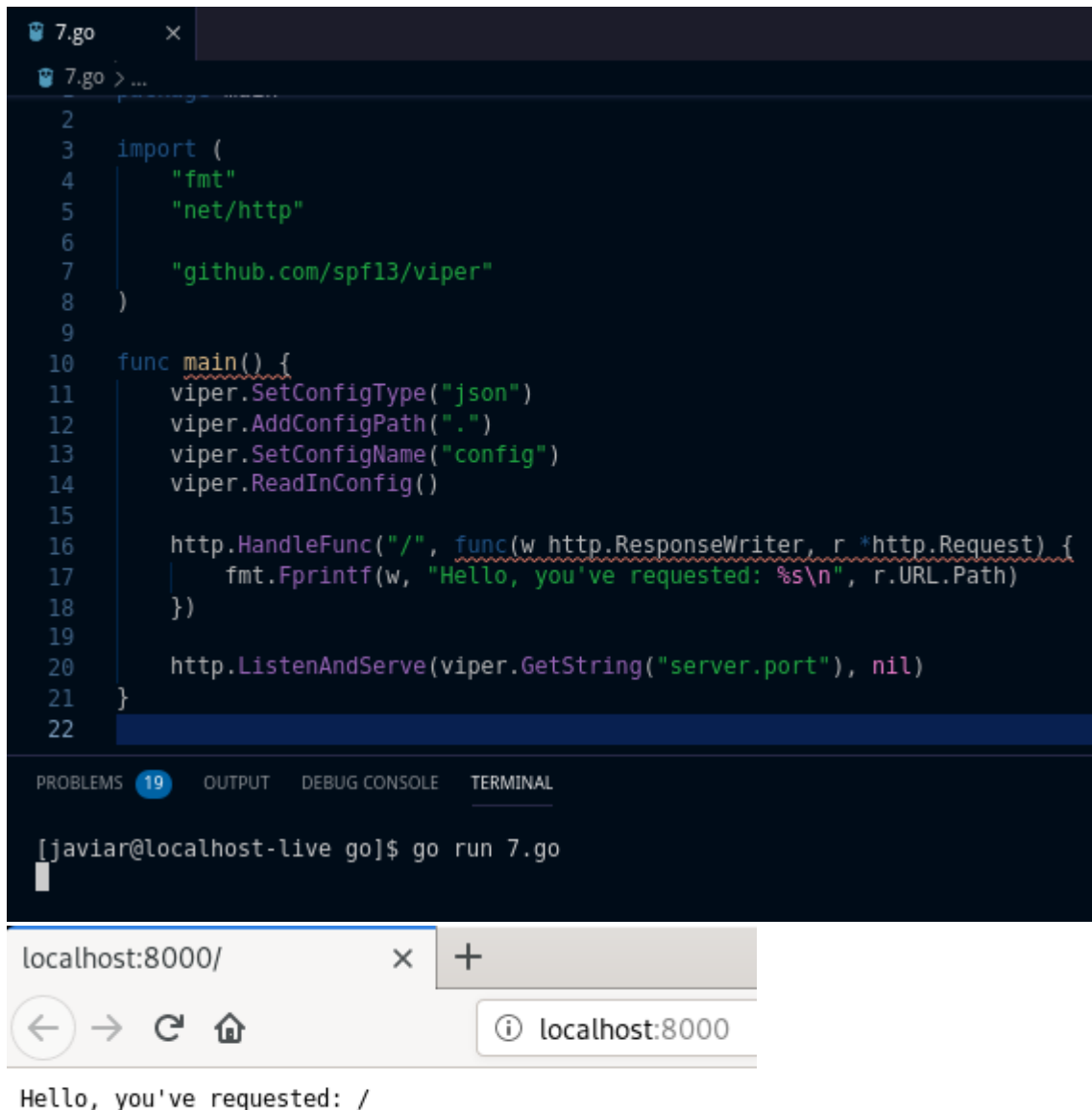
Jawaban Soal No. 6:

Program pada nomor enam merupakan sebuah aplikasi web sederhana dengan menggunakan bahasa Go, berikut adalah hasilnya:



Saat program dijalankan, program akan bertindak sebagai *server* yang menunggu *client* merequest untuk mengakses program (melalui localhost:8000). Saat *client* mengetikkan localhost:8000 ke browser maka program akan menampilkan pesan “Hello, you’ve requested: /”, dimana “/” merupakan halaman awal dari program. Angka 8000 merupakan port yang digunakan program, perubahan pada nilai port maka program bisa tidak terakses.

Jawaban Soal No. 7:



```
7.go
7.go > ...

2
3 import (
4     "fmt"
5     "net/http"
6
7     "github.com/spfl3/viper"
8 )
9
10 func main() {
11     viper.SetConfigType("json")
12     viper.AddConfigPath(".")
13     viper.SetConfigName("config")
14     viper.ReadInConfig()
15
16     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
17         fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
18     })
19
20     http.ListenAndServe(viper.GetString("server.port"), nil)
21 }
22
```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL

```
[javiar@localhost-live go]$ go run 7.go
```

localhost:8000/ x +

localhost:8000

Hello, you've requested: /

Konfigurasi program dilakukan dengan menggunakan file `config.json` yang didalamnya terdapat tipe data `server` dan atributnya `port` (":8000"). Dalam menggunakan viper, perlu dijabarkan tipe data yang menjadi file konfigurasi (sintaks `viper.SetConfigType()`), path file konfigurasi (`viper.AddConfigPath()`), dan nama file konfigurasi (`viper.SetConfigName()`). Lalu, file konfigurasi dibaca dengan sintaks `viper.ReadInConfig()`. Fungsi `http` bawaan bahasa Go mengambil nomor port dari file `config.json` dengan menggunakan viper (sintaks `viper.GetString("server.port")`), yang akan mengembalikan nilai `port` dari tipe data `server` didalam file `config.json`.