Telkom University	**Tugas 5** **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: **I Putu Surya Baratha** ……	**NIM:** **1301188566** ……	**Nilai:** ……
Nama Mahasiswa: **M. Risdham Nur A. P.** ……	**NIM:** **1301188603** ……	**Nilai:** ……
Nama Mahasiswa: **Sella Tresnasari** ……	**NIM:** **1301188565** ……	**Nilai:** ……

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. Lakukan instalasi flatbuffer (https://google.github.io/flatbuffers/) untuk mengerjakan salah satu tugas pada modul ini.
6. Kumpulkan link repository github tersebut sebagai tanda bahwa anda mengerjakan tugas modul ini.
7. Link repository harus berbeda untuk setiap tugasnya. Buatlah markdown yang rapi disetiap repository tugas yang anda kumpulkan.
8. Printscreen program harus dari desktop kelompok anda sendiri, dan harus dari linux yang sudah diinstall. Jika tidak, maka harus mengulang pengerjaan tugasnya.
9. Jangan lupa untuk menuliskan NAMA dan NIM pada laporan.
10. Laporan berbentuk PDF dan dikumpulkan pada link repository github beserta kodenya.
11. Walaupun tugas berkelompok tapi pengumpulan link github harus individu, jika tidak mengumpulkan maka dianggap tidak mengerjakan.

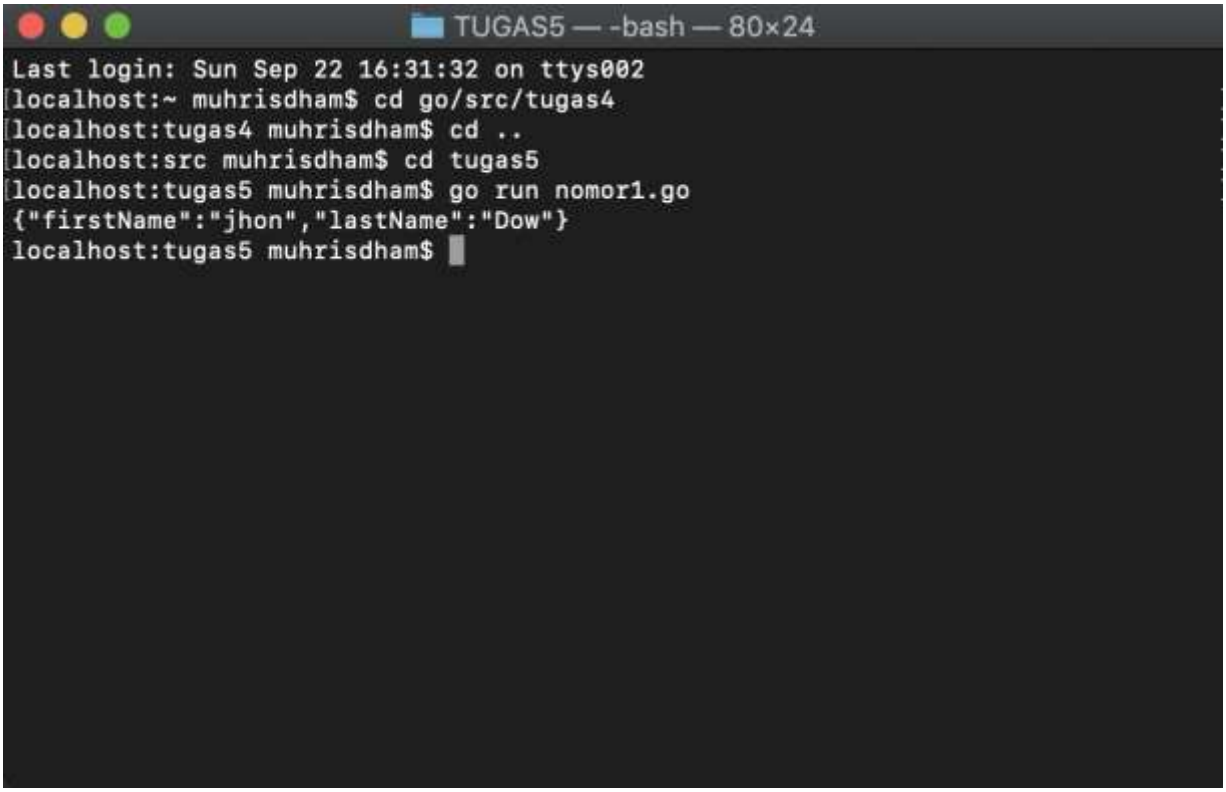
Nama:	NIM:	Nilai:

Soal No 1 (JSON Marshal)

```
package main

import (
    "encoding/json"
    "fmt"
)

type Person struct {
    FirstName string `json:"firstName"`
    LastName  string `json:"lastName"`
}

func main() {
    bytes, err := json.Marshal(Person{
        FirstName: "John",
        LastName:  "Dow",
    })
    if err != nil {
        panic(err)
    }

    fmt.Println(string(bytes))
}
```

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

Jawaban:



Fungsi Marshal() dapat mengambil apa pun, yang didalam GO berarti antarmuka kosong dan mengembalikan sepotong byte serta kesalahan. Jika Marshal() gagal membuat serialisasi nilai input, akan mengembalikan kesalahan non-nil. Marshal() memiliki beberapa batasan ketat, yaitu:

Nama:	NIM:	Nilai:

- Kunci peta harus berupa string.
- Nilai peta harus jenis yang dapat diserialkan oleh paket json.
- Jenis berikut tidak didukung: Channel, kompleks, dan fungsi.
- Struktur data siklik yang tidak didukung.
- Pointer akan dikodekan (dan kemudian didekodekan) sebagai nilai-nilai yang mereka tunjuk (atau 'null' jika pointer nol).

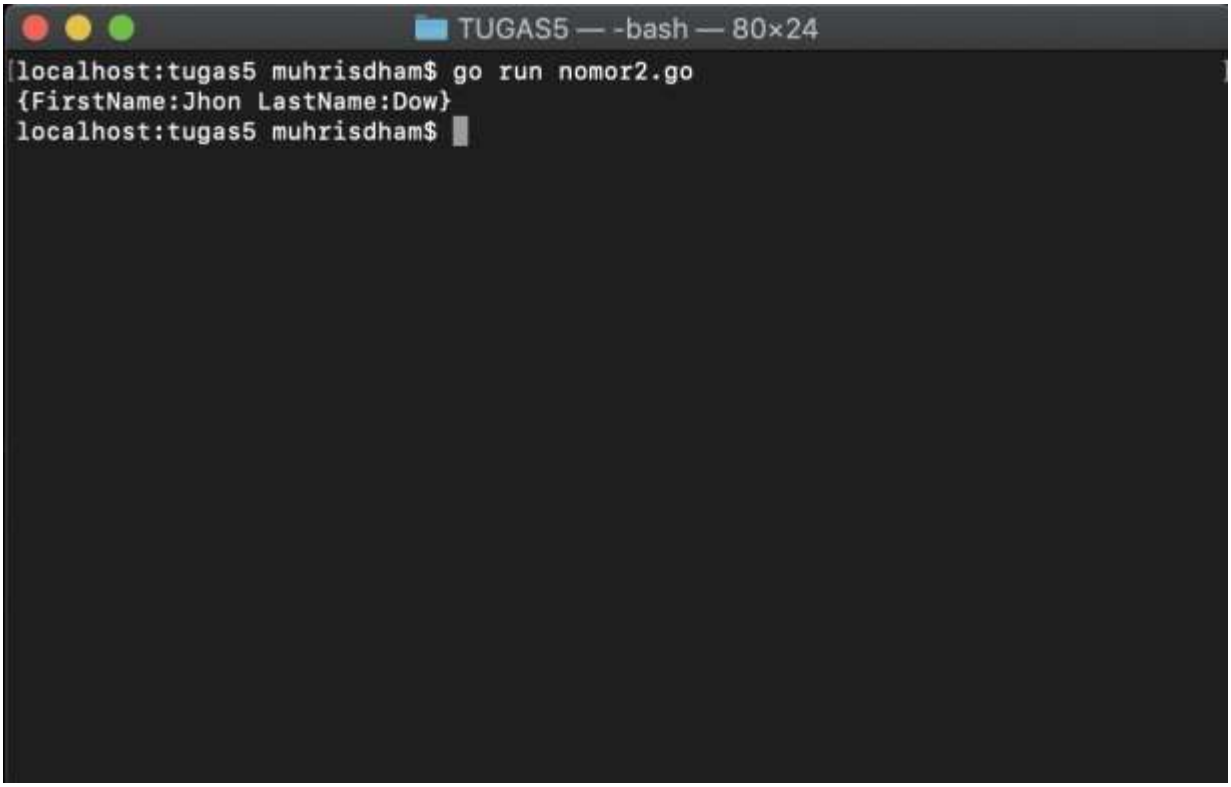
Nama:	NIM:	Nilai:

Soal No 2 (JSON Unmarshal)

```
package main

import (
        "encoding/json"
        "fmt"
)

type Person struct {
        FirstName string `json:"firstName"`
        LastName  string `json:"lastName"`
}

func main() {
        in := `{"firstName":"John","lastName":"Dow"}`
        bytes := []byte(in)

        var p Person
        err := json.Unmarshal(bytes, &p)
        if err != nil {
                panic(err)
        }

        fmt.Printf("%+v", p)
}
```

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

Jawaban:



Fungsi Marshal() dapat mengambil apa pun, yang didalam GO berarti antarmuka kosong dan mengembalikan sepotong byte serta kesalahan. Jika Marshal() gagal membuat serialisasi nilai input, akan mengembalikan kesalahan non-nil. Marshal() memiliki beberapa batasan ketat,

Nama:	NIM:	Nilai:

yaitu: • Kunci peta harus berupa string. • Nilai peta harus jenis yang dapat diserialkan oleh paket json. • Jenis berikut tidak didukung: Channel, kompleks, dan fungsi. • Struktur data siklik yang tidak didukung. • Pointer akan dikodekan (dan kemudian didekodekan) sebagai nilai-nilai yang mereka tunjuk (atau 'null' jika pointer nol).

Cara Kerja :

- Di Go, data json dituliskan sebagai string. Dengan menggunakan json.Unmarshal, json string bisa dikonversi menjadi bentuk objek, entah itu dalam bentuk map[string]interface{} ataupun objek struct.
- Program diatas adalah contoh cara decoding json ke bentuk objek. Pertama import package yang dibutuhkan, lalu siapkan struct Person. Struct Person ini nantinya digunakan untuk membuat variabel baru penampung hasil decode json string. Proses decode sendiri dilakukan lewat fungsi json.Unmarshal().
- Fungsi unmarshal hanya menerima data json dalam bentuk []byte, maka dari itu data json string pada kode di atas di-casting terlebih dahulu ke tipe []byte sebelum dipergunakan pada fungsi unmarshal. Juga, perlu diperhatikan, argument ke-2 fungsi unmarshal harus diisi dengan pointer dari objek yang nantinya akan menampung hasilnya. Jika kita perhatikan lagi, pada struct Person, salah satu property-nya yaitu firstName memiliki tag json:"FirstName". Tag tersebut digunakan untuk mapping informasi json ke property yang bersangkutan.
- Data json yang akan diparsing memiliki 2 property yaitu FirstName dan LastName. Dengan menambahkan tag json, maka property FirstName & LastName struct akan secara cerdas menampung data json property FirstName & LastName.

Soal No 3 (Flatbuffer dan Protocol Buffer)

Jalankan program pada repository github berikut: https://github.com/jonog/grpc-flatbuffers-example

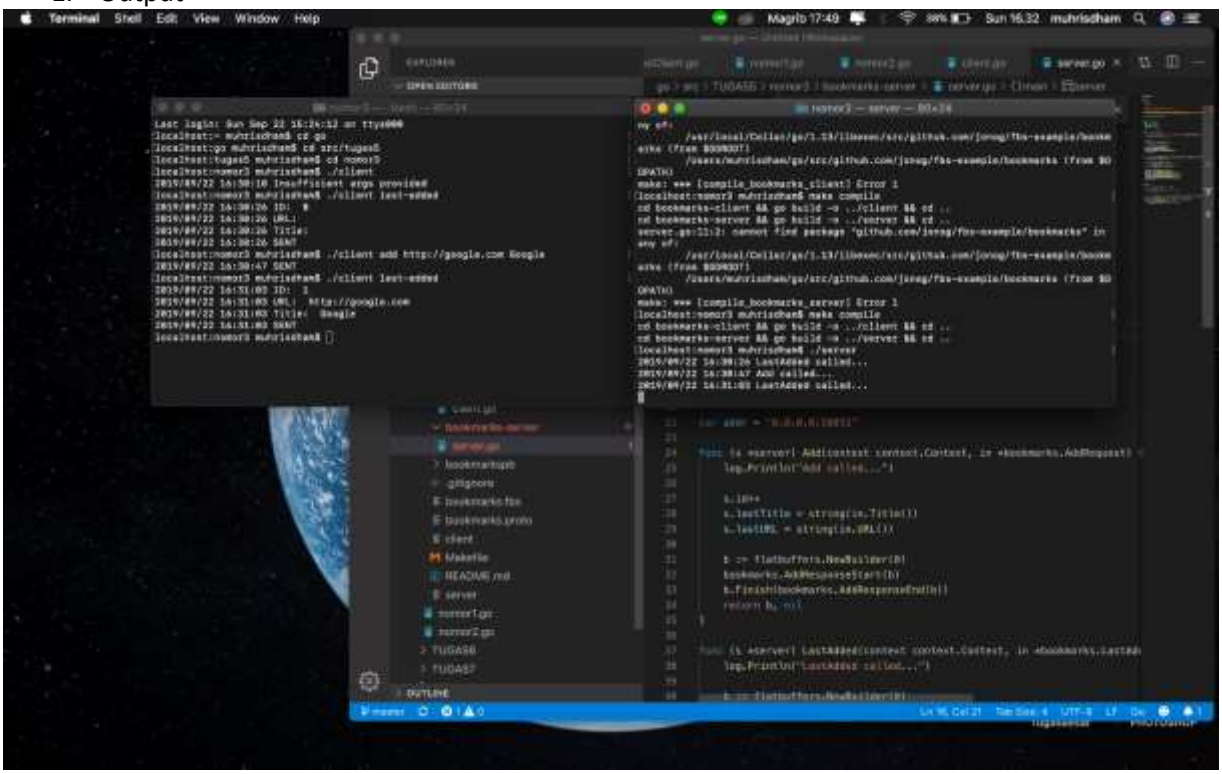
Berikan analisis berupa:

1. Apakah outputnya (berikan printscreen)!
2. Jelaskan cara kerjanya dan buatlah diagram FSMnya!
3. Analisis perbedaan dari protocol buffer dan flatbuffer!

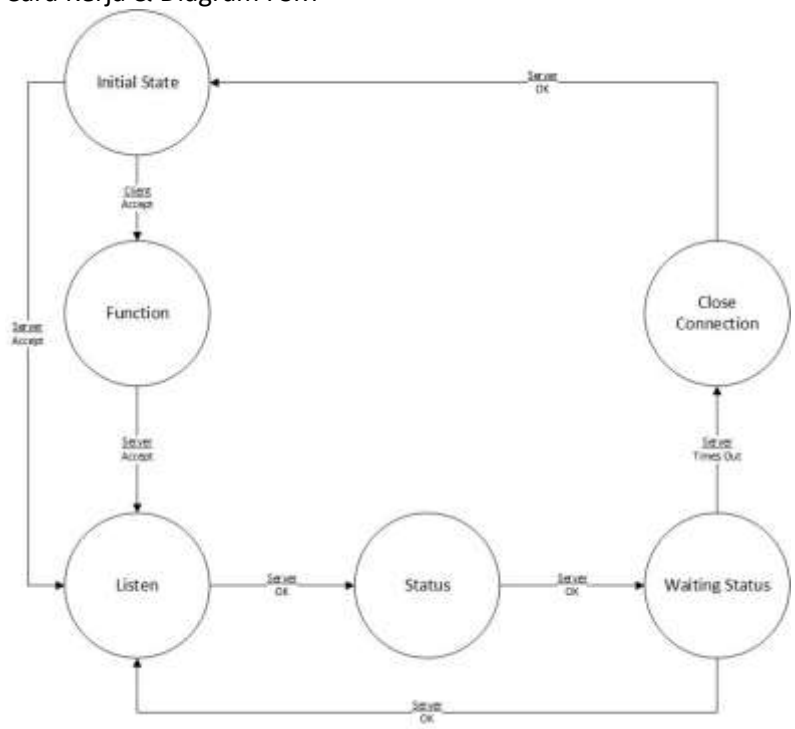
Nama:	NIM:	Nilai:

Jawaban:

1. Output



2. Cara Kerja & Diagram FSM



Nama:	NIM:	Nilai:

3. Analisis perbedaan dari protocol buffer dan flatbuffer
   - Protocol Buffer
     Protocol Buffers adalah metode untuk serialisasi data terstruktur, yang dibuat oleh Google. Protobuf cocok digunakan pada aplikasi yang berkomunikasi dengan aplikasi lain. Protobuf bisa dipakai di banyak platform, contoh: komunikasi antara aplikasi mobile iOS dan Golang Web Service, bisa menggunakan protobuf.
     Protobuf hanya bertugas di bagian serialisasi data saja, untuk komunikasi antar service atau antar aplikasi sendiri menggunakan gRPC.
   - Flatbuffer
     Perbedaan utama antara protobuf dan buffer datar adalah bahwa tidak perlu melakukan deserialisasi seluruh data pada yang terakhir sebelum mengakses objek. Ini membuat buffer rata bertanggung jawab atas kasus penggunaan yang memiliki banyak data. Ini juga mengkonsumsi memori jauh lebih sedikit daripada protobuf.
     Kode juga merupakan urutan besarnya lebih kecil dan mendukung lebih banyak fitur skema (mis. Serikat jenis dalam XML)
     Flatbuffers juga menderita kelemahan yang sama dengan protobuf karena kurangnya representasi yang dapat dibaca manusia.