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:	**NIM:**	**Nilai:**
Javiar Fasyah	1301164477	……
Nama Mahasiswa:	**NIM:**	**Nilai:**
Fahrur Rozi Syarbini	1301164213	……
Nama Mahasiswa:	**NIM:**	**Nilai:**
Hilmi Triandi N	1301164286	……

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)

```go
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:

Nama:	NIM:	Nilai:

Struct Person diinisialisasi pada variabel bytes dengan atribut nama depan (Firstname) “John” dan nama belakang (Lastname) “Dow”. Kemudian, bytes diserialisasikan kedalam bentuk JSON dan dicetak menghasilkan keluaran struct Person dalam bentuk JSON ({“firstname”:”John”,”lastname”:”Dow”}).

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:

Nama:	NIM:	Nilai:

Struct Person dalam bentuk JSON di-assign ke variabel in, yang kemudian variabel in menjadi nilai untuk variabel bytes dalam tipe data []byte. Kemudian, variabel bytes di decode dari bentuk JSON menjadi bentuk struct Person dan ditampung pada variabel p. Sehinga, keluaran yang dihasilkan dari mencetak variabel p adalah {Firstname:John Lastname:Dow}.

Soal No 3 (Flatbuffer dan Protocol Buffer)

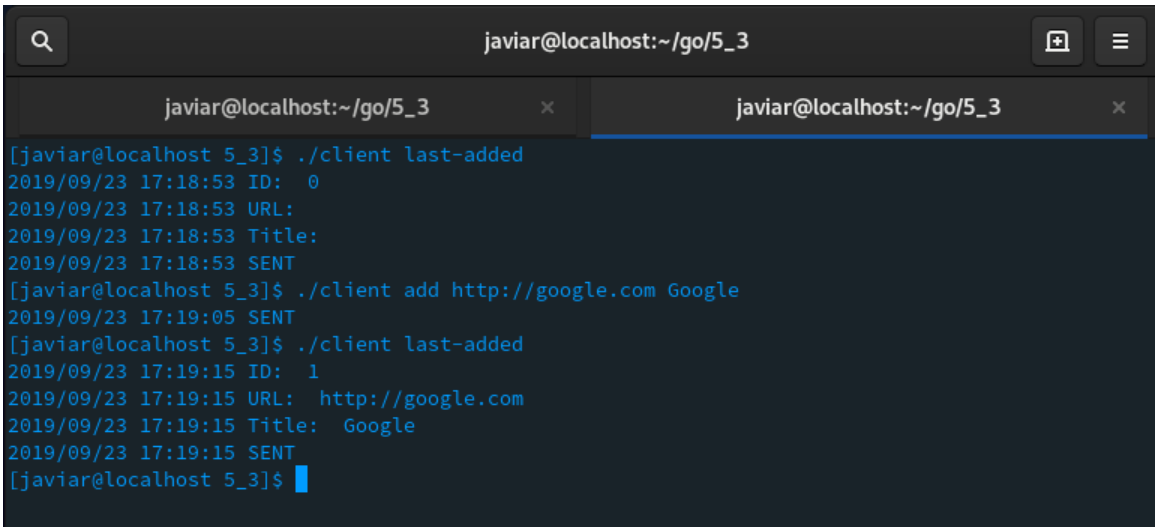
Jalankan program pada repository github berikut: https://github.com/jonog/grpc-flatbuffers-example

Berikan analisis berupa:

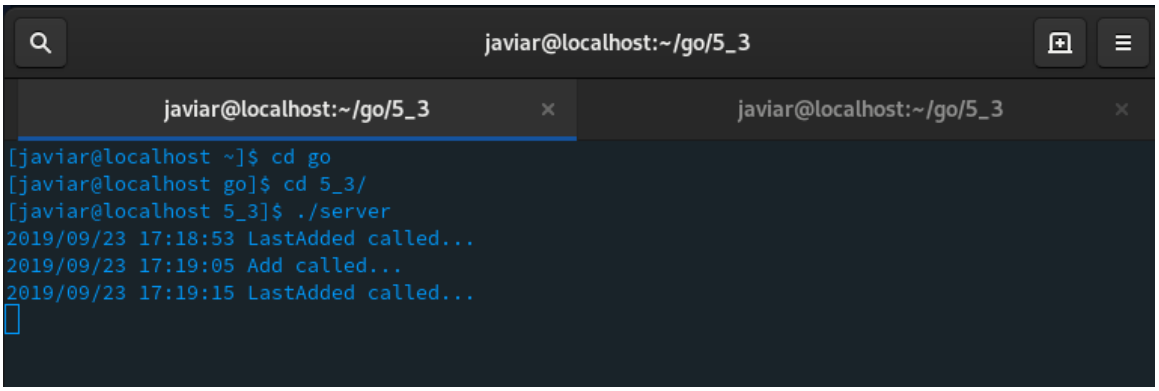
1. Apakah outputnya (berikan printscreen)!
2. Jelaskan cara kerjanya dan buatlah diagram FSMnya!
3. Analisis perbedaan dari protocol buffer dan flatbuffer!

Nama:	NIM:	Nilai:

Jawaban:
Sisi client:



Sisi server:



Perbedaan antara protocol buffer dengan flatbuffer adalah pada representasi in-memory dan wire format-nya. Protocol buffer memisahkan representasi in-memory dengan wire protocolnya (butuh parsing dan serialsiasi), sementara flatbuffer tidak memisahkan keduanya (serialisasi terjadi disaat pembuatan objek flatbuffer pada representasi in-memory). Hal yang sama terjadi saat objek protocol buffer ingin dikembalikan ke asalnya, terjadi parsing dan deserialisasi lagi. Namun pada flatbuffer, yang dtiampilkan adalah pointer kepada objek flatbuffer yang ada.

Nama:	NIM:	Nilai: