

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

Nama Mahasiswa: Mazaya Z D	NIM: 1301154508	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. Disarankan untuk 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. Kumpulkan link repository github tersebut sebagai tanda bahwa anda mengerjakan tugas modul ini.
5. Buatlah Virtual Machine (VM) dengan virtualbox (<https://www.virtualbox.org/wiki/Downloads>). Gunakan operating system Centos (<https://www.centos.org/>) pada VM tersebut. VM yang sudah anda buat akan digunakan sebagai server FTP.

Nama:	NIM:	Nilai:
-------	------	--------

Soal No 1 (Host Lookup)

```

/* ResolveIP
*/

package main

import (
    "fmt"
    "net"
    "os"
)

func main() {
    if len(os.Args) != 2 {
        fmt.Fprintf(os.Stderr, "Usage: %s hostname\n", os.Args[0])
        fmt.Println("Usage: ", os.Args[0], "hostname")
        os.Exit(1)
    }
    name := os.Args[1]

    addr, err := net.ResolveIPAddr("ip", name)
    if err != nil {
        fmt.Println("Resolution error", err.Error())
        os.Exit(1)
    }

    fmt.Println("Resolved address is ", addr.String())
    os.Exit(0)
}

```

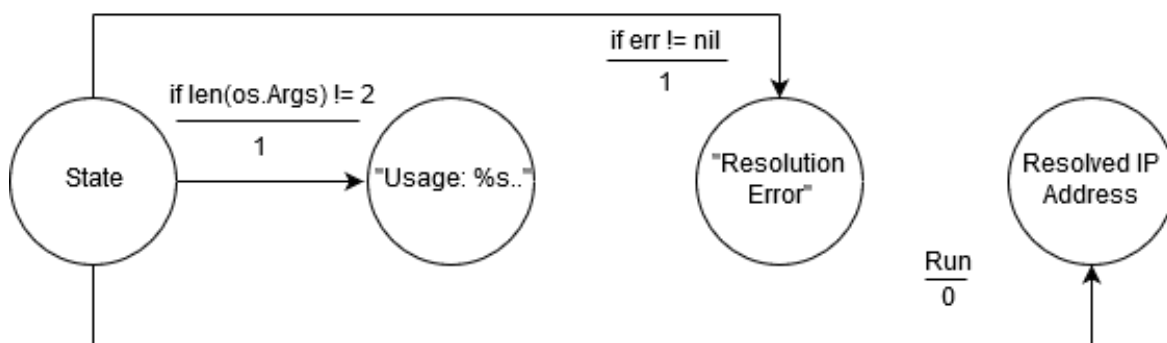
Jalankan program diatas (`go run ResolveIP.go www.google.com`), apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya menggunakan diagram FSM!

Jawaban:

```

[mazayaazd@localhost gp_projects]$ go run ResolveIP.go www.google.com
Resolved address is 74.125.24.104

```



Nama:	NIM:	Nilai:
-------	------	--------

Soal No 2 (Service Lookup)

```

/* LookupPort
*/

package main

import (
    "fmt"
    "net"
    "os"
)

func main() {
    if len(os.Args) != 3 {
        fmt.Fprintf(os.Stderr,
            "Usage: %s network-type service\n",
            os.Args[0])
        os.Exit(1)
    }
    networkType := os.Args[1]
    service := os.Args[2]

    port, err := net.LookupPort(networkType, service)
    if err != nil {
        fmt.Println("Error: ", err.Error())
        os.Exit(2)
    }

    fmt.Println("Service port ", port)
    os.Exit(0)
}

```

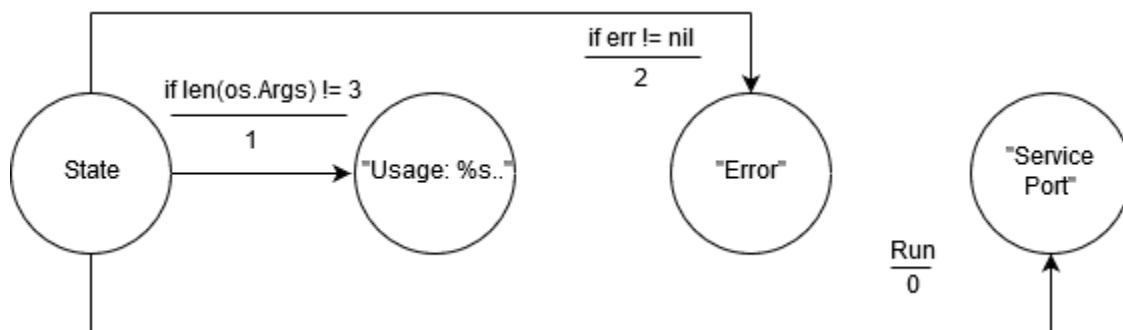
Jalankan program diatas (go run LookupPort.go tcp telnet), apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya menggunakan diagram FSM!

Jawaban:

```

[mazayaazd@localhost gp_projects]$ go run LookupPort.go tcp telnet
Service port 23
[mazayaazd@localhost gp_projects]$

```



Nama:	NIM:	Nilai:
-------	------	--------

Soal No 3 (TCP Client)

```

/* GetHeadInfo
*/
package main

import (
    "fmt"
    "io/ioutil"
    "net"
    "os"
)

func main() {
    if len(os.Args) != 2 {
        fmt.Fprintf(os.Stderr, "Usage: %s host:port ", os.Args[0])
        os.Exit(1)
    }
    service := os.Args[1]

    tcpAddr, err := net.ResolveTCPAddr("tcp4", service)
    checkError(err)

    conn, err := net.DialTCP("tcp", nil, tcpAddr)
    checkError(err)

    _, err = conn.Write([]byte("HEAD / HTTP/1.0\r\n\r\n"))
    checkError(err)

    result, err := ioutil.ReadAll(conn)
    checkError(err)

    fmt.Println(string(result))

    os.Exit(0)
}

func checkError(err error) {
    if err != nil {
        fmt.Fprintf(os.Stderr, "Fatal error: %s", err.Error())
        os.Exit(1)
    }
}

```

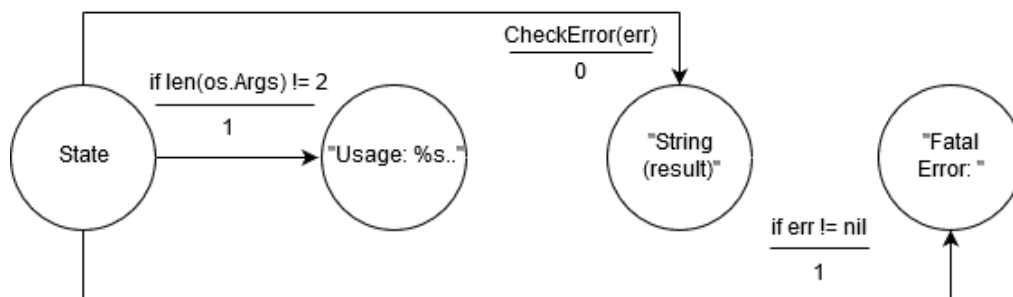
Jalankan program diatas (go run GetHeadInfo.go http://www.google.com:80), apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya menggunakan diagram FSM!

Jawaban:

```

[mazayaazd@localhost gp_projects]$ go run GetHeadInfo.go http://www.google.com:80
Fatal error: address http://www.google.com:80: too many colons in addressexit status 1
[mazayaazd@localhost gp_projects]$

```



Nama:	NIM:	Nilai:
-------	------	--------

Soal No 4 (Raw Sockets and the IPConn Type)

```

/* Ping
*/
package main

import (
    "bytes"
    "fmt"
    "io"
    "net"
    "os"
)

// change this to my own IP address or set to 0.0.0.0
const myIPAddress = "192.168.1.2"
const ipv4HeaderSize = 20

func main() {
    if len(os.Args) != 2 {
        fmt.Println("Usage: ", os.Args[0], "host")
        os.Exit(1)
    }

    localAddr, err := net.ResolveIPAddr("ip4", myIPAddress)
    if err != nil {
        fmt.Println("Resolution error", err.Error())
        os.Exit(1)
    }

    remoteAddr, err := net.ResolveIPAddr("ip4", os.Args[1])
    if err != nil {
        fmt.Println("Resolution error", err.Error())
        os.Exit(1)
    }

    conn, err := net.DialIP("ip4:icmp", localAddr, remoteAddr)
    checkError(err)

    var msg [512]byte
    msg[0] = 8 // echo
    msg[1] = 0 // code 0
    msg[2] = 0 // checksum, fix later
    msg[3] = 0 // checksum, fix later
    msg[4] = 0 // identifier[0]
    msg[5] = 13 // identifier[1] (arbitrary)
    msg[6] = 0 // sequence[0]
    msg[7] = 37 // sequence[1] (arbitrary)
    len := 8

    // now fix checksum bytes
    check := checksum(msg[0:len])
    msg[2] = byte(check >> 8)
    msg[3] = byte(check & 255)

```

Nama:	NIM:	Nilai:
-------	------	--------

```

        // send the message
        _, err = conn.Write(msg[0:len])
        checkError(err)

        fmt.Print("Message sent:  ")
        for n := 0; n < 8; n++ {
            fmt.Print(" ", msg[n])
        }
        fmt.Println()

        // receive a reply
        size, err2 := conn.Read(msg[0:])
        checkError(err2)

        fmt.Print("Message received:")
        for n := ipv4HeaderSize; n < size; n++ {
            fmt.Print(" ", msg[n])
        }
        fmt.Println()
        os.Exit(0)
    }

    func checkSum(msg []byte) uint16 {
        sum := 0

        // assume even for now
        for n := 0; n < len(msg); n += 2 {
            sum += int(msg[n])*256 + int(msg[n+1])
        }
        sum = (sum >> 16) + (sum & 0xffff)
        sum += (sum >> 16)
        var answer uint16 = uint16(^sum)
        return answer
    }

    func checkError(err error) {
        if err != nil {
            fmt.Fprintf(os.Stderr, "Fatal error: %s", err.Error())
            os.Exit(1)
        }
    }

    func readFully(conn net.Conn) ([]byte, error) {
        defer conn.Close()

        result := bytes.NewBuffer(nil)
        var buf [512]byte
        for {
            n, err := conn.Read(buf[0:])
            result.Write(buf[0:n])
            if err != nil {
                if err == io.EOF {
                    break
                }
            }
            return nil, err
        }
        return result.Bytes(), nil
    }

```

Nama:	NIM:	Nilai:
-------	------	--------

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

Jawaban:

Bagaimana mengirim pesan ping ke host. Ping menggunakan perintah "echo" dari protokol ICMP. Ini adalah protokol yang berorientasi byte, di mana klien mengirimkan aliran byte ke host lain, dan host membalas. formatnya adalah:

- Byte pertama adalah 8, berdiri untuk pesan gema
- Byte kedua adalah nol
- Byte ketiga dan keempat adalah checksum pada seluruh pesan
- Byte kelima dan keenam adalah pengidentifikasi arbitrer
- Ketujuh dan delapan byte adalah nomor urut sembarang
- Sisa paket adalah data pengguna

Program berikut akan menyiapkan koneksi IP, mengirim permintaan ping ke host dan mendapatkan balasan.

Untuk mencapai tujuan, paket data masing masing aplikasi mempunyai alamat PC yg dituju dan alamat aplikasi yg dituju. Alamat PC berupa IP adress dan alamat aplikasi (proses) berupa nomor port (port yang dimaksud disini bukan port yg ada di microcontroller). Kombinasi IP adress dan nomor port disebut socket.

```
[mazayaazd@localhost gp_projects]$ sudo go run Ping.go 192.168.88.142
[sudo] password for mazayaazd:
Message sent: 8 0 247 205 0 13 0 37
Mesaage received: 8 0 247 205 0 13 0 37
[mazayaazd@localhost gp_projects]$
```

Nama:	NIM:	Nilai:
-------	------	--------

Soal No 5 (Multi-Threaded Server)

```

package main

import (
    "bufio"
    "fmt"
    "net"
)

func check(err error, message string) {
    if err != nil {
        panic(err)
    }
    fmt.Printf("%s\n", message)
}

func main() {
    ln, err := net.Listen("tcp", ":8080")
    check(err, "Server is ready.")

    for {
        conn, err := ln.Accept()
        check(err, "Accepted connection.")

        go func() {
            buf := bufio.NewReader(conn)

            for {
                name, err := buf.ReadString('\n')

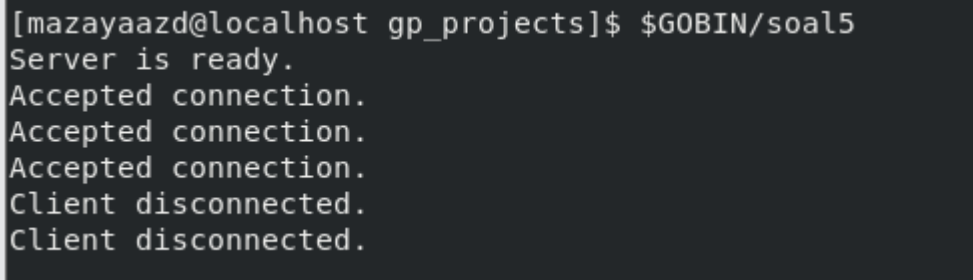
                if err != nil {
                    fmt.Printf("Client disconnected.\n")
                    break
                }

                conn.Write([]byte("Hello, " + name))
            }
        }()
    }
}

```

Jalankan program diatas di dalam virtual box yang sudah anda buat, kemudian lakukan telnet ke port 8080 dalam jumlah yang banyak secara bersamaan, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

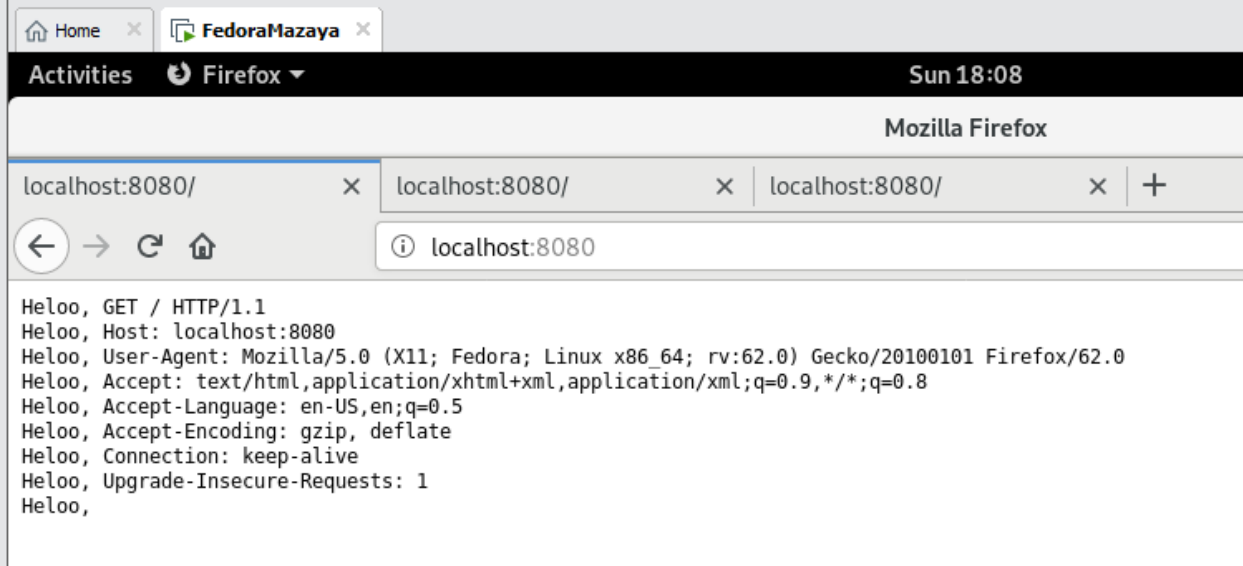


```

[mazayaazd@localhost gp_projects]$ $GOBIN/soal5
Server is ready.
Accepted connection.
Accepted connection.
Accepted connection.
Client disconnected.
Client disconnected.

```


Nama:	NIM:	Nilai:
-------	------	--------



Activities Firefox Sun 18:08

Mozilla Firefox

localhost:8080/ × localhost:8080/ × localhost:8080/ × +

localhost:8080

```

Hello, GET / HTTP/1.1
Hello, Host: localhost:8080
Hello, User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Hello, Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Hello, Accept-Language: en-US,en;q=0.5
Hello, Accept-Encoding: gzip, deflate
Hello, Connection: keep-alive
Hello, Upgrade-Insecure-Requests: 1
Hello,

```

1. Listening to the TCP connection
2. Reading the Request
3. A Closer Look at an HTTP Request
4. Writing a response
5. Validating the Request and Selectively Responding

Ketika client menutup tab yang sedang berjalan, maka response yang diberika 'client disconnected'

Nama:	NIM:	Nilai:
-------	------	--------

Soal No 6 (Multi-Threaded Server)

```

package main

import (
    "bufio"
    "fmt"
    "net"
    "time"
)

func check(err error, message string) {
    if err != nil {
        panic(err)
    }
    fmt.Printf("%s\n", message)
}

type ClientJob struct {
    name string
    conn net.Conn
}

func generateResponses(clientJobs chan ClientJob) {
    for {
        // Wait for the next job to come off the queue.
        clientJob := <-clientJobs

        // Do something thats keeps the CPU busy for a whole second.
        for start := time.Now(); time.Now().Sub(start) < time.Second; {
        }

        // Send back the response.
        clientJob.conn.Write([]byte("Hello, " + clientJob.name))
    }
}

func main() {
    clientJobs := make(chan ClientJob)
    go generateResponses(clientJobs)

    ln, err := net.Listen("tcp", ":8080")
    check(err, "Server is ready.")

    for {
        conn, err := ln.Accept()
        check(err, "Accepted connection.")

        go func() {
            buf := bufio.NewReader(conn)

            for {
                name, err := buf.ReadString('\n')

                if err != nil {
                    fmt.Printf("Client disconnected.\n")
                    break
                }

                clientJobs <- ClientJob{name, conn}
            }
        }()
    }
}

```

Nama:	NIM:	Nilai:
-------	------	--------

Jalankan program diatas di dalam virtual box yang sudah anda buat, kemudian lakukan telnet ke port 8080 dalam jumlah yang banyak secara bersamaan, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

```
[mazayaazd@localhost gp_projects]$ $GOBIN/soal6
Server is ready.
Accepted connection.
Accepted connection.
Accepted connection.
Client disconnected.
```

1. Learn a bit about TCP and HTTP.
2. Listen for TCP connections on a socket.
3. Parse a small number of HTTP requests.
4. Create a proper HTTP response.
5. Improve the throughput of our server with a thread pool.