SVEUČILIŠTE U ZAGREBU

Fakultet elektrotehnike i računarstva

Programski jezik GO

2. Domaća zadaća

ak. god. 2020/2021

UPUTE:

1. Zadaci se predaju na https://fergo.dev/
2. Prvi put se registrirajte na https://fergo.dev/signup (pazite na email i JMBAG)
 - ubuduće se u sustav prijavljujete s emailom i lozinkom koju ste napisali u ovom koraku
 - sustav nije povezan s ferweb / AAIEdu i sl. sustavima, nemojte koristiti istu lozinku!
 - ako zaboravite lozinku, javite se nekom od nastavnika.
3. Odaberite 2. zadaću i zadatak koji želite predati
 Tab „Zadatak“ sadrži tekst zadatka i kratke upute, u tab „Rješenje“ pišete svoje rješenje.
 (Demo zadatak služi za isprobavanje funkcionalnosti sustava i ne boduje se.)
4. Copy/paste ili natipkajte rješenje zadatka u formular
5. Kliknuti na "Predaja zadatka"
6. Ponoviti korake 3-5 za ostale zadatke

NAPOMENE

- predaje se **cijeli program** (package, imports, funkcija main i funkcija koju trebate dopisati)
- isti zadatak se može predati i **više puta**, pamti se samo **zadnja predaja**
- imena funkcija **MORAJU BITI ISTA** kao u zadatku

1. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati funkciju *totalCost* koja vraća ukupnu cijenu artikala. Ako neki artikl premašuje dopuštenu vrijednost kalorija, aplikacija mora vratiti grešku *errHighCalorieFood.*

Rješenje predajete u sljedećem formatu:

```go
package main

import (
    "errors"
    "fmt"
)

var errHighCalorieFood = errors.New("high calorie food not allowed")

type ShopItem struct {
    Name     string
    Price    int
    Callory  int64
    Quantity int
}

func sendData(shopItems chan ShopItem) {
    shopList := []ShopItem{
        {"Hosomaki Ginger", 54, 250, 1},
        // {"Mlijecna cokolada", 8, 600, 3}, // ako se otkomentira ispisuje grešku
        {
            Name:     "Banana",
            Price:    10,
            Callory:  150,
            Quantity: 6,
        },
    }
    for _, item := range shopList {
        shopItems <- item
    }
    close(shopItems)
}

func main() {
    num := make(chan ShopItem, 1)
    go sendData(num)
    mostCommon, err := totalCost(300, num)
    if err != nil {
        fmt.Println(err)
    } else {
        fmt.Println("Total cost is", mostCommon) // rezultat primjera: mostCommon = 114
    }
}

func totalCost(maxCalorie int64, numbers chan ShopItem) (int, error) {
    // tijelo funkcije
}
```

NAPOMENA: Predaje se cijeli program, tj. cijeli sadržaj gornjeg okvira (definicije paketa, importi, varijable, funkcija main, funkcija totalCost)

Primjer:

za rezultat iz zadatka programa mostCommon varijabla će bit postavljena na 114, a err na nil.

2. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati funkciju *add*. Funkcija prima parametre kao *url* argumente, npr: *http://localhost:8080/add/?n1=10&n2=100* i kao rezultat vraća strukturu *webResponseAdd* u JSON formatu. Ako nedostaju argumenti ili su u krivom formatu (string) potrebno je vratiti status code 400 (http.StatusBadRequest)

Rješenje predajete u sljedećem formatu:

```
package main

import (
    "encoding/json"
    "net/http"
    "strconv"
    "strings"
)

type webResponseAdd struct {
    Number1 int `json:"n1"` // input number1
    Number2 int `json:"n2"` // input number2
    Result  int `json:"r"`  // result of number1 + number2
}

func main() {
    http.HandleFunc("/add/", add)
    http.ListenAndServe(":8080", nil)
}

func add(w http.ResponseWriter, r *http.Request) {
    // tijelo funkcije
}
```

NAPOMENE:
Predaje se cijeli program, tj. cijeli sadržaj gornjeg okvira (definicije paketa, importi, varijable, funkcija main, funkcija add)

Primjer:

http://localhost:8080/add/?n1=10&n2=100 vraća {"n1":10,"n2":100,"r":110}

http://localhost:8080/add/?n1=10 vraća samo status code 400

3. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati funkciju minFromChann koja preko kanala prima ulazni niz brojeva i vraća najmanji poslani broj pomnožen sa zadnje tri znamenke Vašeg JMBAG-a.

Potrebno je obratiti pozornost na sljedeće:

- Ako na kanalu ne dolazi ništa dulje od 1 sekunde, potrebno je vratiti errTimeout
- Ako nema ni jednog poslanog broja, a kanal je zatvoren, potrebno je vratiti grešku errNoData.

Rješenje predajete u sljedećem formatu:

```go
package main

import (
    "errors"
    "fmt"
    "time"
)

var (
    errNoData  = errors.New("no data")
    errTimeout = errors.New("timeout")
)

func sendData(numbers chan int) {
    num := []int{1, 2, 3, 4, 5, 1, 2, 3, 1}
    for _, v := range num {
        numbers <- v
    }
    close(numbers)
}

func main() {
    num := make(chan int, 1)
    go sendData(num)
    mostCommon, err := minFromChann(num)
    if err != nil {
        fmt.Println(err)
    } else {
        fmt.Println(mostCommon)
    }
}

func minFromChann(numbers chan int) (int, error) {
    // tijelo funkcije
}
```

NAPOMENA: Predaje se cijeli program, tj. cijeli sadržaj gornjeg okvira (definicije paketa, importi, funkcija main, funkcija minFromChann)

PRIMJER:
u slučaju da je JMBAG 0011111012, rezultat se množi sa 12.