SVEUČILIŠTE U ZAGREBU

Fakultet elektrotehnike i računarstva

Programski jezik GO

1. 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 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, a kasnije i za ostale zadaće ☺

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 dvije funkcije: funkciju **`totalCallory`** koja vraća ukupni broj kalorija svih artikala u shopping listi, te funkciju **`mostHealthy`** koja vraća najzdraviji artikl (ili više njih ako svi imaju isti broj kalorija).

```
package main

import (
    "errors"
    "fmt"
)

var errNoData = errors.New("no data in list")

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

func main() {
    shopList := []ShopItem{
        {"Hosomaki Ginger", 54, 250, 1},
        {"Mlijecna cokolada", 8, 600, 3},
        {"Banana", 10, 150, 6},
    }
    tCal := totalCallory(shopList)
    fmt.Println(tCal) // 2950
    best, err := mostHealthy(shopList)
    fmt.Println(best) // [{Banana 10 150 6}]
    fmt.Println(err)       // nil
}

func totalCallory(shoppingList []ShopItem) (total int) {
    // tijelo funkcije
}

func mostHealthy(shoppingList []ShopItem) (items []ShopItem, err error) {
    // tijelo funkcije
}
```

NAPOMENA: Predaje se cijeli program, tj. cijeli sadržaj gornjeg okvira (definicije paketa, importi, funkcija main, funkcije totalCallory i mostHealthy)

Primjer:

Za shopping listu iz zadatka funkcija **`mostHealthy`** vraća

```
[{Banana 10 150 6}]
```

a funkcija totalCallory vraća

```
2950
```

2. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati funkciju **`countWords`** koja vraća ukupni broj različitih riječi u rečenici, te mapu kojoj je ključ riječ, a vrijednost broj ponavljanja te riječi. Osim riječi, u tekstu se mogu pojaviti točka, zarez i višestruki razmaci.

```go
package main

import (
    "fmt"
    "strings"
)

func main() {
    numbers := "Lorem ipsum dolor sit amet"
    num, occurrence := countWords(numbers)
    fmt.Println(num, occurrence) // 5 map[amet:1 dolor:1 ipsum:1 lorem:1 sit:1]
}

func countWords(sentence string) (int, map[string]int) {
    // tijelo funkcije
}
```

NAPOMENE:
- zarezi, razmaci i točke su jedini dodatni znakovi koja se pojavljuju osim samih riječi
- funkcija ne razlikuje velika i mala slova, ***Lorem*** i ***lorem*** su ista riječ
- možete napisati i dodatne funkcije, ako je potrebno

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

Primjer:

Za rečenicu iz zadatka funkcija **`countWords`** vraća

```
5 map[amet:1 dolor:1 ipsum:1 lorem:1 sit:1]
```

Za rečenicu "Lorem, lorem" rezultat bi bio

```
1 map[lorem:2]
```

3. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati funkciju **`sumarise`** koja nadodaje/zbraja ulazni parametar u varijablu **`sum`**. Pripaziti da rješenje bude moguće korištenjem **go rutina** kao u primjeru main-a.

```go
package main

import (
    "fmt"
    "sync"
)

var (
    wg  sync.WaitGroup
    mux sync.Mutex
    sum int
)

func sumarise(number int) {
    // tijelo funkcije
}

func main() {
    numbers := []int{1, 2, 3, 4, 5, 1, 2, 3}
    wg.Add(len(numbers))
    for _, number := range numbers {
        num := number
        go sumarise(num)
    }
    wg.Wait()
    fmt.Println(sum) // 21
}
```

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

Primjer:

Za numbers listu iz zadatka varijabla **`sum`** će bit postavljena na **`21`**