SVEUČILIŠTE U ZAGREBU

Fakultet elektrotehnike i računarstva

Programski jezik GO

1. Domaća zadaća

ak. god. 2019/2020

UPUTE:

1. Zadaci se predaju na https://fergo.info.tm
2. Upišite svoje ime i prezime, JMBAG i e-mail (ovo je potrebno napraviti samo jednom)
3. Kliknite na prijava (ako planirate zadatke predavati u više navrata, zapamtite svoj **Student Id**)
4. Odaberite zadatak koji želite predati
5. Copy/paste rješenja zadatka u formular
6. Kliknuti na "Predaja X. zadatka"

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 **MORAJ BITI ISTA** kao u zadatku

1. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati funkciju **`counter`** koja uzima *slice* brojeva i kao rješenje vraća *mapu* kojoj su ključevi brojevi, a vrijednost broj ponavljanja ključa u *slice*-u.

```
package main
import "fmt"
func main() {
	numbers := []int{1, 2, 3, 4, 5, 1, 2, 3}
	solution := counter(numbers)
	fmt.Println(solution)
}
func counter(numbers []int) map[int]int {
	// tijelo funkcije
}
```

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

Primjer:

```
numbers := []int{1, 5, 1}
solution := Counter(numbers)
```

Očekivani rezultat:

```
map[int]{ 1: 2, 5: 1, }
```

2. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati funkciju **`sretanBroj`** koja će za ulaznu matricu od m x n jedinstvenih brojeva, vratiti "sretan broj" iz matrice. Sretan broj je element matrice koji je minimalan u svojem retku i maksimalan u svojem stupcu.

Primjeri:
- za matricu [[9, 12][3, 4]] sretan broj je 9.
- za matricu [[3,10,4,2],[9,3,8,7],[15,14,13,12]] sretan broj je 12.

```
package main
import "fmt"
func main() {
    numbers := [][]int{
     []int{9,12},
     []int{3,4},
    }
    solution := sretanBroj(numbers)
    fmt.Println(solution)
}

func sretanBroj (matrica [][]int) int {
    // tijelo funkcije
}
```

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

3. ZADATAK: Potrebno je nadopuniti zadani program, tj. napisati dvije funkcije: funkciju **`mostExpensive`** koja vraća najskuplji artikl (ili više njih), te funkciju **`totalCost`** koja vraća ukupnu cijenu svih artikala u shopping listi.

Primjeri:
- ako je shopping lista prazna, **`mostExpensive`** vraća praznu listu, a **`totalCost`** vraća `0`.
- ako ima više artikala iste (maksimalne) cijene, funckija **`mostExpensive`** vraća sve takve artikle.
- ako je shopping lista nil, **`mostExpensive`** javlja **`fmt.Errorf("no data")`**, a **`totalCost`** vraća `0`.
- za shopping listu iz donjeg programa funkcija **`mostExpensive`** vraća "`mlijeko`", a funkcija **`totalCost`** vraća `31`

```
package main
import "fmt"
type Shopping struct {
      Name string
      Price int
      Quantity int
}

func main() {
      shopList := []Shopping{
          Shopping{"kruh", 8, 2},
          Shopping{"mlijeko", 15, 1}, }
      solution1, err := mostExpensive(shopList)
      fmt.Println(solution1) // mlijeko
      fmt.Println(err) // nil
      solution2 := totalCost(shopList)
      fmt.Println(solution2) // 31
}
func mostExpensive(shopList []Shopping) (item []Shopping, err error) {
    // tijelo funkcije
    // pripaziti: kad se može dogoditi pogreška?
}

func totalCost(shopList []Shopping) (total int) {
    // tijelo funkcije
}
```

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