Midterm, 10.12.2018 Meno,priezvisko:

Pravidlá:

- **Ļas 100 minút**
- **môžete používať akékoľvek ale len Vami prinesené materialy,**
- **nie je možné zdieľať akékoľvek pomôcky a materiály,**
- **kolektívne riešenia sa netolerujú,**
- **každý príklad riešte na papieri obsahujúcom jeho zadanie,**
- **midterm obsahuje 5 príkladov spolu za 24 bodov.**

1. [4 body – Maticový]

Nech `Matica=[[Int]]` reprezentuje obdĺžnikovú celoĺ íselnú maticu.

Úlohy:

- **[1 bod]** Definujte funkciu `nulovyRiadok::Matica->Bool`, ktorá platí, ak matica obsahuje nulový riadok.
- **[1 bod]** Definujte funkciu `nulovyStlpec::Matica->Bool`, ktorá platí, ak matica obsahuje nulový stĺpec.

[2 body] Definujte funkcie `zrotuj90::Matica->Matica` a `zrotuj180::Matica->Matica`, ktoré zrotujú maticu reprezentovanú ako zoznam riadkov o 90 (proti smeru hod.ruľ iľ iek) a 180 stupňov.

Príklad:

```
zrotuj90 [[1,2,3],[4,5,6]]=[[3,6],[2,5],[1,4]]
zrotuj180 [[1,2,3],[4,5,6]]=[[6,5,4],[3,2,1]]
```

-------- v prípade potreby riešte túto úlohu na zadnú stranu **TOHOTO listu**----------

2.[6 bodov - priemer]

Váġený priemer neprázdnej mnoġiny dát x_i s váhami w_i je definovaný na obrázku dole.
Váha w_i teda predstavuje dôleġitosŠ s akou sa do priemeru poļíta hodnota x_i. Aritmetický priemer je len ġpeciálny prípad, ak $w_i = 1$. Pre zjednoduġenie, v celom príklade predpokladajte, ġe vektory váh a hodnôt sú rovnako dlhé, a váhy sú nenulové. Definujme dva základné typy a testovacie konġtanty, vĐavo sú vektory, vpravo matice:

```
type Vector = [Float]
datas   :: Vector
datas   = [1.0, 2.0]
weights :: Vector
weights = [0.1, 0.2]
```

```
type Matrix   = [Vector]
datass  :: Matrix
datass  = [ [1.0, 2.0], [3.0, 4.0] ]
weightss :: Matrix
weightss = [ [ 0.1, 0.2], [ 0.3, 0.4] ]
```

Úlohy:

- **[1 bod]** Definujte funkciu **wavg::Vector->Vector->Float**, vypoļíta váġený priemer z vektora hodnôt a vektora váh, teda **wavg datas weights = (1*0.1 + 2*0.2)/(0.1+0.2) = 1.6666…**
  [1 bod] Definujte funkciu **dataWeights :: Vector->Vector->[(Float, Float)]**, vráti zoznam dvojíc hodnota s váhou, teda **dataWeights datas weights = [(1.0,0.1),(2.0,0.2)].**
- **[2 body]** Definujte funkciu **wavgF::Vector->Vector->Float**, ktorá vypoļíta váġený priemer, ale teraz musíte pouġiŠniektorú zo schém foldr/foldl, a teda trafiŠsa do definície v tvare **wavgF = ? foldr/foldl ? ?**, resp. **wavgF ds ws = ? foldr/foldl ? ? ? ds ws**
  Hint: Pozor, foldl/foldr beġí cez jeden zoznam, nikdy nie cez dva zároveŘ. Chce to tomu pomôcŠ..
- **[2 body]** Definujte funkciu **wavgM::Matrix->Matrix->Float**, ktorá vypoļíta váġený priemer prvkov matice. Ak vaġa definícia predpokladá, ġe matica má rovnako dlhé riadky, uveĎte to. Ak funguje aj pre maticu s rôzne dlhými riadkami, uveĎte to. Omyl má penalizáciu 0.5 b. Príklad: **wavgM datass weightss = 3.0.**

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i},$$

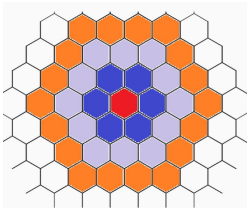
-------- v prípade potreby rieġte túto úlohu na zadnú stranu **TOHOTO listu**----------

3. [3 body ï ġesŠuholníkový]

Ġesšuholníkové ļ ísla sú také, ġe z nich je moġné vytvoriš pravidelný ġesšuholník podča obrázku dole. Sú to **1**, **7** (=1+6), **19** (=1+6+12), **37**, é

Úlohy:

- **[2 body]** Definujte nekoneč ný zoznam ġesšuholníkových ļ ísel **`sest::[Int]`**, teda **`take 10 sest = [1,7,19,37,61,91,127,169,217,271].`**
- **[1 bod]** Definujte funkciu **`obvod::Int->Int`**, ktorá pre ġesšuholníkové ļ íslo vráti dĺžku obvodu ġesšuholníka, teda **`obvod 7 = 6, obvod 19 = 12.`** Pre iné ako ġesšuholníkové ļ ísla nie je funkcia definovaná.



-------- v prípade potreby rieġte túto úlohu na zadnú stranu **TOHOTO listu**----------

4. [4 body – stromový]

Binárny strom s celoč íselnými hodnotami vo vnútorných vrcholoch je definovaný takto:

```
data Tree = Nil | Node Tree Int Tree  deriving(Show, Eq)
t :: Tree
t = Node (Node ((Node Nil 3 Nil)) 1 (Node Nil 3 Nil)) 5 (Node (Node Nil 6 Nil) 7 (Node Nil 9 Nil))
```

Úlohy:

- **[2 body]** Definujte test **uplny::Tree->Bool**, č i strom je úplný binárny strom. Jedna z definícií úplného stromu môže byť taká, že vetky jeho listy sú rovnako vzdialené od koreňa. Zamyslite sa nad inou definíciou, kým začnete programovať

- Funkcia **toList::Tree->[Int]**, definovaná takto

```
toList  Nil = [0]
toList  (Node left value right) = 99 : value : ((toList left) ++ (toList right))
```

ktorá prevedie strom do zoznamu č ísel. 99 je moja obľubená konštanta. Funguje to takto:

```
toList (Node Nil 1 Nil) = [99,1,0,0]
toList (Node (Node Nil 2 Nil) 1 Nil) = [99,1,99,2,0,0,0]
toList (Node (Node Nil 2 Nil) 1 (Node Nil 3 Nil)) = [99,1,99,2,0,0,99,3,0,0]
```

[2 body] Je možné napísať funkciu, ktorá z tejto reprezentácie zoznamu č ísel vyrobí späť pôvodný strom ? Zrejme nie každý zoznam je obrazom nejakého stromu, napríklad [], [99], [17], [99, 3, 0] nie sú. Pre takéto zoznamy funkcia nie je definovaná. Hodnotí sa vaše zdôvodnenie. Ak nejde strom spätne vytvoriť, napíšte dôvod, resp. príklad zoznamu. Ak to ide, tak definujte funkciu **fromList::[Int]->Tree**, ktorá z obrazu stromu v tvare zoznamu **[Int]** opäť vyrobí pôvodný strom. Platí, že pre ľubovoľné **t::Tree, fromList (toList t) == t**. Pre zoznam, ktorý nie je obrazom stromu, je funkcia nedefinovaná.

-------- v prípade potreby riešte túto úlohu na zadnú stranu **TOHOTO listu**----------

5. [7 bodov ï Go]

Toto je kód z prednáġky ilustrujúci 100 ļ ínskych ġepkárov/agentov, ktorí si posielajú správu typu int, kaġdý Ņalġí k nej pripoļ íta 1. KeĻ dostane správu posledný z nich, vypíġe ju na konzolu.

```
var number = 100
func main() {
    prev := make(chan int)
    first := prev
    go func() { first <- 0 }()
    for i := 0; i < number; i++ {
        next := make(chan int)
        go func(from, to chan int) {
            for {
                to <- 1 + <-from
            }
        }(prev, next)
        prev = next
    }
    fmt.Println(<-prev)
}
```

Do ich správania chceme pridaŠnasledujúce aspekty:

- **[1 bod]** kaġdý ġepkár/agent má svoj vek **v**, ļ o je náhodné ļ íslo **0<=v<90**, a kým sa mu podarí preposlaŠsprávu Ņalej, trvá mu to presne **v** milisekúnd, *hint: rand.Intn(n)*.
- **[1 bod]** novú správu vġdy s nasledujúcim indexom (t.j. 0, 1, 2, 3, 4 é) posielame prvému ġepkárovi/agentovi pravidelne kaġdých 10 sekúnd. Kaġdý ġepkár obdrġanú správu pred jej Ņalġím preposlaním pozmení tak, ġe k jej obsahu pripoļ íta 1000. Preto je vġdy moġné zo správy zistiŠjej pôvodný index (pôvodný obsah) aj poļ et krokov - preposlaní medzi agentami.
- **[1 bod]** keĻ posledný ġepkár/agent s indexom 99 nedostane ġiadnu správu uġ celých 10 sekúnd, protestuje na konzolu (`YOU ARE TOO LATE`).
- **[1 bod]** celá simulácia konļ í po 60 sekundách (`GAME IS OVER`)
- **[3 body]** ġepkár/agent **i neposiela** správu svojmu priamemu následovníkovi **i+1**, ale **náhodne si vyberie** ġepkára/agenta spomedzi vġetkých, a tomu poġle správu. Ten opäŠvyberá náhodného príjemcu. DĜka trasy správy potom nemusí byŠ100 krokov, aj správy nemusia prísŠv poradí, v akom boli odoslané. KeĻ k poslednému ġepkárovi 99 správa dorazí, vypisujte jej index aj poļ et krokov, ktoré ubehla.

Môġete rieġiŠniektoré podúlohy len doplnením kódu. Musí vġak byŠjasné, ļ o platí, ļ o ste doplnili, ļ i vyhodili z pôvodného kódu. V zadaní tieġ jasne oznaļ te, ktoré podúlohy ste podĜa vás vyrieġili. V prípade, ġe rieġite **väļ ġinu podúloh, resp. poslednú**, je rozumnejġie napísaŠvlastný kód (lebo úpravy do pôvodného sú pomerne masívne). Pôvodným kódom sa len inġpirujte... Tu je príklad simulácie, ktorý snáĻ zodpovie vaġe iniciálne otázky.

```
START MESSAGE 0              08:23:57 ← nová správa
STOP MESSAGE 0, STEPS 106    08:24:03
START MESSAGE 1              08:24:07 ← nová správa
YOU ARE TOO LATE !           08:24:13   ←posledný už 10 sekúnd nič nedostal
START MESSAGE 2              08:24:17 ← nová správa
STOP MESSAGE 1, STEPS 186    08:24:18
START MESSAGE 3              08:24:27 ← nová správa
YOU ARE TOO LATE !           08:24:28   ←posledný už 10 sekúnd nič nedostal
STOP MESSAGE 3, STEPS 27     08:24:29
STOP MESSAGE 2, STEPS 219    08:24:29
START MESSAGE 4              08:24:37 ← nová správa
YOU ARE TOO LATE !           08:24:39   ←posledný už 10 sekúnd nič nedostal
STOP MESSAGE 4, STEPS 47     08:24:40
START MESSAGE 5              08:24:47 ← nová správa
STOP MESSAGE 5, STEPS 1      08:24:47
START MESSAGE 6              08:24:57 ← nová správa
GAME IS OVER !               08:24:57       ← simulácia skončila po 60s.
```

-------- v prípade potreby rieġte túto úlohu na zadnú stranu **TOHTO listu** ----------

-------- v prípade potreby rieġte túto úlohu na zadnú stranu **TOHOTO listu** ----------