Midterm, 18.11.2014 Meno,priezvisko:

Pravidlá:

- **čas 90 minút**
- **možete používať akékoľvek ale len Vami prinesené materialy,**
- **knihy, poznámky, listingy, …**
- **nie je možné používať nič elektronické,**
- **nie je možné zdieľať akékoľvek pomôcky a materialy,**
- **kolektívne riešenia sa netolerujú,**
- **každý príklad riešte na papieri obsahujúcom jeho zadanie,**
- **midterm obsahuje 4 príklady spolu za 22 bodov.**

1) **[6 bodov – Maticový]** Trojuholníková matica je skrátený zápis diagonálne symetrickej štvorcovej matice, ktorej druhá polovica nad hlavnou uhlopriečkou je rovnaká. Príklad je o symetrických maticiach,teda o maticiach symetrických podľa hlavnej diagonály,inak povedané m=transpose m. Ich prvky vieme reprezentovať pomocou typu **`Matica=[[Int]]`** v tvare „trojuholníka“, teda

`[[`a_{11}**`],`**
`[`a_{21}**`, `**a_{22}**`],`**
`…`
`[`a_{n1}**`, `**a_{n2}**`, … , `**a_{nn}**`]]::Matica.`**

Príklad:

```
[ [1],  -- jednotková matica 3x3
  [0, 1],
  [0, 0, 1],
  [0, 0, 0, 1] ] ::Matica.
```

- **[1bod]** Definujte funkciu **`jeTrojuholnikova::Matica->Bool`**, ktorá zistí, či hodnota typu **`Matica`** má trojuholníkový tvar (teda riadky dĺžky postupne 1..n pre nejaké n), ako v príklade hore.
- **[1bod]** Definujte funkciu **`jednotkova::Int->Matica`**, ktorá pre volanie **`jednotkova n`** vráti jednotkovú maticu trojuholníkového tvaru veľkosti n x n, ako v príklade hore (**`jednotkova 3).`**
  [2body] Súčet dvoch trojuholníkových (symetrických) matíc je trojuholníková (symetrická) matica, preto definujte funkciu **`sucet::Matica->Matica->Matica`**, ktorá sčíta dve rovnako veľké trojuholníkové matice do výslednej trojuholníkovej.
  Príklad: **`sucet(jednotkova 3) (jednotkova 3)=[[2],[0,2],[0,0,2]].`**
- **[2body]** Definujte funkciu **`nulovyRiadok::Matica->Bool`**, ktorá zistí z trojuholníkovej reprezentácie matice, či jej **pôvodná štvorcová symetrická** matica obsahuje nulový riadok. Pozor, nepýtame sa, či trojuholníková matica má nulový riadok, ale či ho má celá pôvodná štvorcová matica. Kým tento rozdiel nepochopíte, študujte príklad, resp. sa pýtajte. Príklad:

```
nulovyRiadok [ [1], [2,3], [4,5,6] ] == False
nulovyRiadok [ [0], [0,3], [0,5,6] ] == True
nulovyRiadok [ [1], [2,0], [4,0,6] ] == False
nulovyRiadok [ [1], [0,0], [4,0,6] ] == True
nulovyRiadok [ [1], [2,3], [0,0,0] ] == True
```

Test, či ste pochopili: ak diagonálne symetrická matica má nulový riadok, má aj nulový stĺpec. Áno-Nie?

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

2) [7 bodov – foldový]

V tomto zadaní je **^** funkcia mocnina, teda **(\x -> \y -> x^y**), a teda **2^3==8** a tiež **3^2==9**.

- **[1 bod]** vypočítajte výrazy (jeden je **foldr** a druhý **foldl**). Určite, ktorý je väčší, resp.rovnaké
  foldr (^) 3 [3,3]
  foldl (^) 3 [3,3,3]

V každej ďalšej časti úlohy si stačí vybrať jednu z verzií **foldr** alebo **foldl**, podľa preferencií.

- **[2body]** do definície funkcie `expList::[Int]->(Int->Int)` doplňte výrazy za symbol **?** tak, aby funkcia pre vstupný zoznam **$[a_1,a_2,\ldots,a_n]$ vrátila funkciu \x->(((x^a_1)^a_2)…^a_n)**.
  expList xs = foldr ? ? xs alebo **expList xs = foldl ? ? xs.**
- **[2 body]** definujte **funkciu tripartita::[t]->([t],[t],[t]),** ktorá rozdelí zoznam na tri približne rovnaké zoznamy. *Rozdelí* znamená, že všetky prvky pôvodného zoznamu sa nachádzajú v niektorom z troch zoznamov, a naopak. *Približne rovnaké* znamená, že ich dĺžky sa líšia najviac o jeden. **2 body** za riešenie v tvare
  tripartita xs = foldr ? ? xs, alebo **tripartita xs = foldl ? ? xs**
  iné riešenie len 1 bod.
- **[2 body]** definujte funkciu **pivot::(Ord t)=>t->[t]->([t],[t],[t]),** ktorá rozdelí zoznam na trojicu troch zoznamov prvkov menších, rovných a väčších jako pivot p, a to v tvare:
  pivot p xs = foldr ? ? xs, alebo **pivot p xs = foldl ? ? xs**.

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

3) [5 bodov – BVS]

Toto je definícia parametrického binárneho stromu s hodnotami typu `t` vo vnútorných vrcholoch.

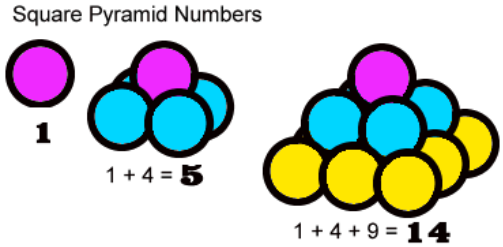
```
data BVS t = Node (BVS t) t (BVS t) | Nil deriving (Show, Eq)
```

Ak vieme hodnoty typu **`t`** porovnávať, tak má zmysel sa baviť, či tento strom spĺňa vlastnosti binárneho vyhľadávacieho stromu. V príklade predpokladáme, že vstupy do definovaných funkcií spĺňajú túto sémantickú vlastnosť, preto ju neoverujte a len definujte:

- **[2 body]** funkciu **`fromTo::(Ord t)=>t->t->(BVS t)->[t]`**, ktorá pre volanie **`fromTo a b tree`** vráti usporiadaný zoznam prvkov binárneho vyhľadávacieho stromu väčších ako **`a`** a menších ako **`b`**. Myslite na efektívnosť aj v prípade, ak je strom veľký a interval (**`a;b`**) malý.
- **[3 body]** funkciu **`jeVyvazeny::(BVS t)->Bool`**, ktorá zistí, či binárny strom je vyvážený, t.j. výšky oboch podstromov ľubovoľného vrchola sa líšia najviac o 1.

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

4) **[4 body – pyramídový]**
a) [2body] Skutočné pyramídy sú štvorboké ihlany – podstavou je štvorec, hore končia špicom. Preto skutočne pyramídové čísla sú také, ktoré sa dajú napísať v tvare súčtu $1^2+2^2+3^2+ ... n^2$. Definujte nekonečný zoznam skutočne pyramídových čísel **`sp::[Int]`**, ktorého začiatok má tvar `[1,5,14,30,55,…]`.
b) [2body] Definujte funkciu **`jeSP::Int->Boolean`**, ktorá zistí, či číslo je skutočne pyramídové číslo, príklad: **`jeSP 14==True,jeSP 15==False`**.



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