## Ez az utolsó gyakorlati óra

Pluszminusz: Az év végén 1 +-ra kell állni.

Maximum 3 HF hiányozhat, akinek több hiányzik, annak be kell mutatnia párat.

Évfolyam ZH eredményei fent vannak a BE-AD rendszerben. Mivel sok embernek jól sikerült (kétlem!) a ponthatárok nem fognak lejjebb menni. (Nekem 13 ponttal ötös.)

Pót ZH: Most pénteken ugyan úgy a lovardában, de 16:00-tól. A számon kért anyag több lesz, szükség lesz a múlt heti type és a ma vett algebrai adattípusokra is.

Géptermi vizsga 23-án, kedden. A két jegy átlaga lesz a félévi eredmény. Aki kétes jegyre áll, annak a HF-jei, szorgalmijai, pluszminuszai döntik el a jegyét.

Vizsgára már lehet jelentkezni. Update: Jelentkeztem. (Programozás vizsgával már nem ütközik.)

Folytatást fel lehet venni Speci-ként:

* Haladó Haskell. Párhuzamosság, monád (mellékhatásos számítás)
* Erlang: Mohó kiértékelésű nyelv, érdekes kommunikációs funkciói vannak.

Projektek:

* Erlang refaktorálás
* Hálózati adatforgalom, router programozás Haskell-ben.
* Szoftver(…) labor 4 féléves.

# Származtatott típusok (newtype)

Nem szinonimát hozunk létre, hanem egy teljesen új típust.

A newtype kulcsszóval egy új típust hozhatunk létre korábbi típusokból.

**newtype** Name = N String

Megadunk:

* egy új típusnevet: $Name$
* egy konstruktornevet: $N$
* és egy létező típust: $String$

Az $N$ konstruktor tesz különbséget a $Name$ és a $String$ típus között.

# Algebrai adattípusok (data)

## Ordering

A Prelude-ben így definiálták az Ordering típust:

**data** Ordering

 = LT

 | EQ

 | GT

 **deriving** (Eq, Ord, Show, Data, Typeable)

## Algebrai típusdefiníció egy konstruktorral

**data** IPoint = P Int Int

 **deriving** (Eq, Ord, Show, Data, Typeable)

* $data$ kulcsszó
* $IPoint$ típuskonstruktor
* $P$ konstruktor
* paraméterek

Használat: P 0 0 :: IPoint

Mintaillesztés és konstrukció:

mirrorO :: IPoint -> IPoint

mirrorO (P x y) = P (-x) (-y)

## Eltolás

$\left(P x y\right)$-t be kell zárójelezni, különben 3 extra paraméternek hinné.

translate :: (Int, Int) -> IPoint -> IPoint

translate (tx,ty) (P x y) = P (x+tx) (y+ty)

## Irány típus (Jobbra-balra)

**data** Direction

 = L

 | R

 **deriving** (Eq, Ord, Show, Data, Typeable)

## L-ek száma

numOfLs = length . filter (==L)

numOfLs ls = length $ filter (==L) ls

## Irány tükrözése

Mintaillesztéssel:

mirror L=R

mirror R=L

## Háromértékű válasz

**data** Answer

 = No

 | Maybe

 | Yes

 **deriving** (Eq, Ord, Show)

## Logikai ÉS háromértékűre

No &&& **\_** = No

**\_** &&& No = No

Maybe &&& **\_** = Maybe

**\_** &&& Maybe = Maybe

**\_** &&& **\_** = Yes

### Rövidebben

x &&& y = min x y

### Még rövidebben

(&&&)=min

### Logikai VAGY

(|||)=max

## Égtájak

Kihagyjuk. Valószínűleg jó megoldásom:

**data** CardinalPoint

 =East

 |West

 |North

 |South

 **deriving** (Eq, Ord, Show)

## Bitek

**data** Bit

 = Zero

 | One

 **deriving** (Eq, Ord, Show)

## Bináris számok

**type** Binary = [Bit]

## Következő bináris szám

Nagyon zavaró, hogy fordított sorrendben van. El is rontottam először.

plus1 []=[One]

plus1 (Zero:xs)=One:xs

plus1 (One:xs)=Zero:plus1 xs

## Bináris összeadás

add [] ys = ys

add xs [] = xs

add (Zero:xs) (y:ys) = y:add xs ys

add (x:xs) (Zero:ys) = x:add xs ys

add (One:xs) (One:ys) = Zero:add (plus1 xs) ys

## A Bool típus

A Bool típus definíciója a Prelude-ben:

**data** Bool

 = False

 | True

 **deriving** (Eq, Ord, Show, Read, Enum)

## Összetett algebrai típusok

**data** Color

 = Named String

 | RGB Int Int Int

 | HSV Float Float Float

 **deriving** (Show)

# Paraméterezett algebrai adattípusok

## Maybe

**data** Maybe a

 = Nothing

 | Just a

 **deriving** (Eq, Ord, Show, Typeable, Data)

safeDiv :: Integral a => a -> a -> Maybe a

## Biztonságos osztás

### Hivatalos megoldás

safeDiv a 0 = Nothing

safeDiv a b = Just(div a b)

### Enyém

safeDiv a b

 |b==0=Nothing

 |otherwise=Just $ div a b

A hivatalos jobban tetszik.

## Biztonságos head

safeHead [] = Nothing

safeHead (x:**\_**) = Just(x)

Lehetett volna safeHead l=Just(head l) is.

## Either

**data** Either a b

 = Left a

 | Right b

 **deriving** (Eq, Ord, Show)

## Párok

**data** Tuple2 a b = Tuple2 a b

Tuple2 'a' True :: Tuple2 Char Bool

## Hármasok

**data** Tuple3 a b c = Tuple3 a b c

Tuple3 'a' True "xx" :: Tuple3 Char Bool [Char]

## Gyakorlás

<http://pnyf.inf.elte.hu/fp/CodeBreaker.xml>

"A kód típusa" feladattól érdemes megoldani.