Programiranje I: 1. izpit

26. januar 2017

Čas reševanja je 150 minut. **Funkcij v Haskellu ne pozabite opremiti z ustrezno signaturo.** Veliko uspeha!

1. naloga (Aritmetični izrazi, 20 točk)

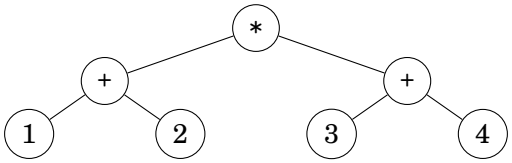
Aritmetični izraz, ki vključuje le cela števila ter njihove vsote in produkte, lahko predstavimo s podatkovnim tipom `Izraz`, ki ga definiramo takole:

```
data Izraz = Samo Integer | Plus Izraz Izraz | Krat Izraz Izraz
```

Aritmetičnemu izrazu `(1 + 2) * (3 * 4)` tako ustreza izraz

```
Krat (Plus (Samo 1) (Samo 2)) (Krat (Samo 3) (Samo 4))
```

ki si ga lahko predstavljamo z drevesom



a) Sestavite funkcijo `izracunaj`, ki izraz evalvira. Na primer:

```
ghci> izracunaj (Samo 1)
1
ghci> izracunaj (Krat (Plus (Samo 1) (Samo 2)) (Samo 3))
9
ghci> izracunaj (Krat (Plus (Samo 1) (Samo 2)) (Krat (Samo 3) (Samo 4)))
36
```

b) Podatkovni tip `Izraz` napravite za primerek razreda tipov `Num`. Vrednosti `abs`, `signum` in `negate` lahko definirate kot `undefined`.

```
ghci> 1 + (Samo 2)
Plus (Samo 1) (Samo 2)
ghci> (1 + 3) * (Plus (Samo 1) (Samo 2))
Krat (Plus (Samo 1) (Samo 3)) (Plus (Samo 1) (Samo 2))
```

2. naloga (Fiksne točke, 20 točk)

Sestavite funkcijo `fiksna`, ki sprejme urejen seznam celih števil $[x_1,\ldots,x_n]$ in v času $O(\log n)$ izračuna najmanjši indeks i, za katerega velja $x_i = i$. Če tak indeks ne obstaja, naj funkcija vrne `None` oziroma `Nothing`. Na primer:

```
>>> fiksna([-4, -2, 0, 2, 4])
4
>>> fiksna([-4, -2, 0, 3, 4])
3
>>> fiksna([-4, -2, 0, 2, 3])
None
```

Nalogo načeloma rešujte v Pythonu, ker boste tako laže dosegli iskano časovno zahtevnost. Če želite uporabiti Haskell, namesto seznamov uporabite tabele, ki jih ponuja modul `Data.Array`.

3. naloga (Orbite, 20 točk)

a) Sestavite funkcijo `orbita`, ki izračuna seznam vseh različnih elementov, ki jih lahko dobimo z zaporedno uporabo dane funkcije na danem elementu. Na primer:

```
ghci> orbita (\x -> mod (x + 2) 10) 13
[13,5,7,9,1,3]
ghci> orbita succ 5
[5,6,7,8,9,10,11,...]
ghci> orbita negate 0
[0]
ghci> orbita negate 1
[1,-1]
```

b) Sestavite funkcijo `generatorji`, ki izračuna najkrajši seznam vseh elementov, ki jih potrebujemo, da z zaporedno uporabo dane funkcije dobimo vse elemente danega seznama. Na primer:

```
ghci> generatorji negate [1,-2,-1]
[1,-2]
ghci> generatorji (\x -> (x + 3) `mod` 10) [1,2,3,4]
[1]
ghci> generatorji (\x -> (x + 2) `mod` 10) [1,2,3,4]
[1,2]
ghci> generatorji (\x -> (x + 2) `mod` 10) [1,2,3,4,5,11]
[2,11]
```

Če je najkrajših seznamov več, lahko funkcija vrne katerega koli.

4. naloga (Vžigalice, 20 točk)

Igralca A in B igrata igro *Vžigalice*. Igralca igrata izmenično, igro prične igrati igralec A. Na začetku igre je pred igralcema kup n vžigalic. Igralec na potezi s kupa odstrani najmanj 3 vžigalice in največ 10 vžigalic. Izgubi tisti igralec, ki ne more izvesti poteze.

a) Sestavite funkcijo `kup`, ki za dano število n učinkovito izračuna, kateri od igralcev ima zmagovalno strategijo. Na primer:

```
>>> kup(2)                    ghci> kup 2
'B'                           "B"
>>> kup(3)                    ghci> kup 3
'A'                           "A"
>>> kup(11)                   ghci> kup 11
'B'                           "B"
```

b) Igro razširimo tako, da igralca igrata z večimi kupi vžigalic. Igralec na potezi si izbere kup in z njega odstrani najmanj 3 in največ 10 vžigalic. Izgubi tisti igralec, ki ne more izvesti poteze. Sestavite funkcijo `kupi`, ki za dan seznam naravnih števil učinkovito izračuna, kateri od igralcev ima zmagovalno strategijo. Na primer:

```
>>> kupi([11])                ghci> kupi [11]
'B'                           "B"
>>> kupi([2, 11])             ghci> kupi [2, 11]
'A'                           "A"
>>> kupi([3, 11])             ghci> kupi [3, 11]
'B'                           "B"
>>> kupi([2, 3, 11])          ghci> kupi [2, 3, 11]
'B'                           "B"
```

Nalogo lahko rešujete v Pythonu ali v Haskellu.