Programiranje I: 2. izpit

11. julij 2017

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

1. naloga (Študenti, 20 točk)

Študente pri predmetu Programiranje 1 lahko predstavimo s tipom `Student`, ki ga definiramo takole:

```
type Student = (Ime, Priimek, VpisnaStevilka, Rezultati)
type Ime = String
type Priimek = String
type VpisnaStevilka = Int
data Rezultati = Neudelezen | Rezultat (Int, Int, Int, Int) deriving (Show)
```

Shranimo si še letnik z naslednjimi tremi študenti:

```
a = ("Ana", "Bertoncelj", 1, Rezultat (10, 20, 15, 2)) :: Student
b = ("Bine", "Cencelj", 2, Rezultat (5, 13, 20, 17)) :: Student
c = ("Cilka", "Drnovsek", 3, Neudelezen) :: Student

type Letnik = [Student]
l = [a, b, c] :: Letnik
```

a) Sestavite funkcijo `vsotaTock`, ki izračuno vsoto točk študenta.

```
ghci> vsotaTock a
Just 47
ghci> vsotaTock c
Nothing
```

b) Sestavite funkcijo `najboljsi`, ki vrne tistega študenta v letniku, ki je dosegel najboljši rezultat.

```
ghci> najboljsi l
("Bine","Cencelj",2,Rezultat (5,13,20,17))
```

2. naloga (Graf, 20 točk)

Podatkovni tip `Graf` predstavimo takole:

```
type Graf = (Vozlisca, Povezave)
type Vozlisca = [Int]
type Povezave = [(Int, Int)]

prazniGraf = ([], []) :: Graf
```

Shranimo si še en primer grafa:

```
v = [1,2,3,4,5] :: Vozlisca
p = [(1,2), (1,3), (2,4), (3,5), (4,5)] :: Povezave
g = (v, p) :: Graf
```

a) Sestavite funkcijo `jeGraf`, ki preveri, če je vhodni graf res graf, torej če so vse naštete povezave le med danimi vozlišči.

```
ghci> jeGraf g
True
ghci> jeGraf ([1,2], [(1,3)])
False
```

b) Podatkovni tip obogatimo do tipa `PobarvaniGraf`, ki ima vsako vozlišče obarvano. Barve predstavimo kot naravna števila.

```
type PobarvaniGraf = (Graf, [Barva])
type Barva = Int
```

Sestavite funkcijo `lepoPobarvaj`, ki sprejme graf in ga lepo pobarva, se pravi vrne pobarvani graf z lastnostjo, da nobeni dve povezani vozlišči nista enake barve. Pri tem naj porabi *čim manj* barv.

```
ghci> lepoPobarvaj ([1,2], [])
(([1,2],[]),[1,1])
ghci> lepoPobarvaj ([1,2], [(1,2)])
(([1,2],[(1,2)]),[1,2])
ghci> lepoPobarvaj g
(([1,2,3,4,5],[(1,2),(1,3),(2,4),(3,5),(4,5)]),[1,2,2,1,3])
```

3. naloga (Tipkanje, 20 točk)

Uslužbenci na veliki slovenski banki morajo ob vsakem plačilu položnice vnesti številko računa v računalnik. Za to uporabljajo standardno tipkovnico, ki ima številke od 0 do 9 razvrščene v eni vrsti. Vsak uslužbenec tipka le s kazalcema in v vsakem trenutku lahko naredi eno od dveh reči:

1. Premakne katerega koli od kazalcev na sosednjo tipko, a le enega naenkrat.
2. Pritisne na tipko, ki je tik pod njegovim kazalcem. Pritisne lahko le eno tipko naenkrat.

Za izvedbo vsakega od gornjih dveh gibov uslužbenec potrebuje eno sekundo. Na začetku ima uslužbenec levi kazalec nad tipko 0, desnega pa nad tipko 9. Številko vnaša od leve proti desni.

Sestavite funkcijo `tipkanje`, ki *učinkovito* izračuna najkrajši čas, ki je potreben, da uslužbenec v računalnik vnese številko.

```
>>> tipkanje("10")
4
>>> tipkanje("107")
7
>>> tipkanje("123456789123456789123456789")
63
>>> tipkanje("4780")
13
>>> tipkanje("01" * 10)
28
```

4. naloga (Najdaljši interval, 20 točk)

Sestavite funkcijo `najdaljsi`, ki kot vhod sprejme seznam `sez` različnih celih števil, vrne pa nabor števil (a, b) iz vhodnega seznama, za kateri velja, da `sez` vsebuje med številoma a in b kar največ elementov, hkrati pa vsi ti elementi ležijo na številskem intervalu $[a, b]$. Če je možnosti več, naj funkcija vrne tisto, pri kateri je par (a, b) leksikografsko zadnji. Funkcija naj deluje v času $O(n \log n)$, pri čemer je n dolžina vhodnega seznama.

```
>>> najdaljsi([0,1,2,-1])
(0,2)
>>> najdaljsi([9,8,7,6,5,4,3,2,1])
(9,9)
>>> najdaljsi([0,1,2,-1,5,7,6,8,-2])
(-1,8)
>>> najdaljsi([-10, 0, -1, 0, 1])
(-1,1)
```