R smer, Programske paradigme 2016/2017, kolokvijum, grupa A

Na *Desktop*-u napraviti direktorijum čije je ime u formatu **ImeIPrezime_BrojIndeksa**. Na primer, **JovanPetrovic_mr14072**. Sve zadatke sačuvati u ovom direktorijumu.Zadatke imenovati sa **1.hs**, **2.hs**, **3.hs**, **4.hs**, **Poeni.hs**, **5.hs**, **Racuni.hs**.
NAPOMENA: *4. i 5. zadatak se rade korišćenjem **dph** paketa. Vektorizovan deo koda pisati u zasebnom modulu koji treba ulinkovati sa glavnim programom koji je zadužen samo za učitavanje podataka za paralelnu obradu i ispis rezultata. Eliminacioni test primeri su obeleženi zvezdicom. Potrebno je da se strogo držite formata ispisa koji je naznačen u zadacima.*

1. Ana uči prirodne brojeve i zanima je na koje sve načine može da razloži dati broj na dva prirodna broja, kao i koliko takvih razlaganja ima. Definisati sledeće funkcije koje pomažu Ani u rešavanju pomenutog problema:

   a) `razlozi :: Int -> [(Int,Int)]` - za dati prirodan broj `n` vraća listu parova `(a,b)` takvih da su `a` i `b` prirodni brojevi iz intervala `[1,n]` i da je zbir `a` i `b` jednak `n`

Primer 1*
```
Poziv: razlozi 3
Izlaz:
 [(1,2),(2,1)]
```
Primer 2
```
Poziv: razlozi 4
Izlaz:
 [(1,3),(2,2),(3,1)]
```
Primer 3
```
Poziv: razlozi 1
Izlaz:
 []
```

   b) `brojR :: Int -> Int` - računa na koliko se različitih načina dati prirodan broj `n` može predstaviti u obliku zbira dva prirodna broja

Primer 1*
```
Poziv: brojR 3
Izlaz:
 2
```
Primer 2
```
Poziv: brojR 4
Izlaz:
 3
```
Primer 3
```
Poziv: brojR 1
Izlaz:
 0
```

2. Aleksa i Luka žele da komuniciraju preko šifrovanih poruka koje predstavljaju listom niski. Aleksa šifruje željene podatke na sledeći način: ukoliko se niska koju šalje sastoji samo od cifara, na njen početak i kraj dodaje karakter `C`, ako se sastoji samo od malih slova, na njen početak i kraj dodaje karakter `S`, a inače na njen početak i kraj dodaje karakter `O`. Definisati sledeće funkcije koje pomažu Aleksi da pošalje Luki šifrovanu poruku:

   a) `broj :: [Char] -> Bool` - za datu nisku proverava da li su svi njeni karakteri cifre

Primer 1*
```
Poziv: broj "123"
Izlaz:
 True
```
Primer 2
```
Poziv: broj ['c','a','o']
Izlaz:
 False
```
Primer 3
```
Poziv: broj "Zdravo123"
Izlaz:
 False
```

   a) `mala :: [Char] -> Bool` - za datu nisku proverava da li su svi njeni karakteri mala slova

Primer 1*
```
Poziv: mala "pozdrav"
Izlaz:
 True
```
Primer 2
```
Poziv: mala ['C','a','o']
Izlaz:
 False
```
Primer 3
```
Poziv: mala "1cao2"
Izlaz:
 False
```

   c) `sifruj :: [[Char]] -> [[Char]]` - datu listu niski transformiše na sledeći način: ukoliko se niska koju šalje sastoji samo od cifara, na njen početak i kraj dodaje karakter `C`, ako se sastoji samo od malih slova, na njen početak i kraj dodaje karakter `S`, a inače na njen početak i kraj dodaje karakter `O`.

Primer 1*
```
Poziv: sifruj ["11","maj","zurka"]
Izlaz:
 ["C11C","SmajS","SzurkaS"]
```
Primer 2
```
Poziv: sifruj ["cao","Cao","caO"]
Izlaz:
 ["ScaoS","OCaoO","OcaOO"]
```
Primer 3
```
Poziv: sifruj ["Poz","sA","casa"]
Izlaz:
 ["OPozO","OsAO","ScasaS"]
```

3. Podaci o cenama artikala u prodavnici su zadati kao `type Cene = [Double]`.

   a) Definisati funkciju `razlika :: Cene -> Cene -> [Double]` koja vraća razliku cena artikala iz dve različite prodavnice. Pretpostaviti da u datim listama, podaci na istim pozicijama odgovaraju istim artiklima i da su liste jednakih dužina.

Okrenite stranu!

Primer 1*

```
Poziv: razlika [120,115,12.23] [119,174.56,89.9]
Izlaz:
  [1.0,-59.56,-77.67]
```

Primer 2

```
Poziv: razlika [10,150,21,62] [11,140,89,74]
Izlaz:
  [-1.0,10.0,-68.0,-12.0]
```

Primer 3

```
Poziv: razlika [12.5,15] [11,14.5]
Izlaz:
  [1.5,0.5]
```

b) Definisati funkciju `dupliraj :: Cene -> Int -> Cene` koja za date podatke o cenama i ceo broj `n` duplira cene svih artikala počev od `n`-tog. Pretpostaviti da je indeks `n` ispravno zadat i da se broji od 0 u listi.

Primer 1*

```
Poziv: dupliraj [12,15.5,120] 2
Izlaz:
  [12,15.5,240.0]
```

Primer 2

```
Poziv: dupliraj [12,15.5,120,17.8] 0
Izlaz:
  [24.0,31.0,240.0,35.6]
```

Primer 3

```
Poziv: dupliraj [5.5,15.2,13.9,20,19,21] 3
Izlaz:
  [5.5,15.2,13.9,40.0,38.0,42.0]
```

c) Definisati funkciju `jeftinija :: Cene -> Cene -> Cene` koja duplira cene svih artikala iz prodavnice u kojoj ima više jeftinijih artikala i vraća novodobijene cene za tu prodavnicu. Pretpostaviti da u datim listama, podaci na istim pozicijama odgovaraju istim artiklima i da su liste jednakih dužina. U slučaju da je broj jeftinijih artikala isti, modifikovati cene za prvu prodavnicu.

Primer 1*

```
Poziv: jeftinija [120,115,12.23]
       [119,174.56,89.9]
Izlaz:
  [240.0,230.0,24.46]
```

Primer 2

```
Poziv: jeftinija [11,140.5,89.9,74]
       [10.5,150,21.23,62]
Izlaz:
  [21.0,300.0,42.46,124.0]
```

Primer 3

```
Poziv: jeftinija [12.5,14] [11,14.5]
Izlaz:
  [25.0,28.0]
```

4. Nakon održanog takmičenja iz matematike treba odrediti ko je osvojio najviše poena. Poeni se unose za svakog takmičara u zasebnoj liniji. Redni broj takmičara odgovara rednom broju linije u kojoj su uneti podaci o njegovim poenima. Napisati program koji pronalazi i ispisuje redni broj takmičara koji je osvojio najviše poena. Ukoliko ima više takvih takmičara, ispisati redni broj prvog učitanog među najboljima. Ne praviti pretpostavku o broju takmičara, poeni se učitavaju sve dok se ne unese EOF. Paralelizovati obradu učitanih podataka korišćenjem **dph** paketa. Vektorizovan deo koda pisati u modulu Poeni.

   Uputstvo: *Linkovanje vektorizovanog koda Poeni.hs i glavnog programa 4.hs, nakon čega pokrećete izvršni fajl a.out:*
   ghc -fvectorise -package dph-lifted-vseg Poeni.hs
   ghc -package dph-lifted-vseg 4.hs
   ghc -threaded -rtsopts -package dph-lifted-vseg Poeni.o 4.o

Primer 1*

```
Poziv: ./a.out +RTS -N4
Ulaz:
  99
  100
  85.50
Izlaz:
  2
```

Primer 2

```
Poziv: ./a.out +RTS -N4
Ulaz:
  56
  21.50
Izlaz:
  1
```

Primer 3

```
Poziv: ./a.out +RTS -N4
Ulaz:
  56
  74
  74
Izlaz:
  2
```

5. U datoteci `racuni` se u zasebnim linijama nalaze cene artikala koje je neki kupac pazario u prodavnici. Redni broj računa odgovara rednom broju linije u kojoj su podaci o kupljenim artiklima sa tog računa. Napisati program koji ispisuje koliko iznosi ukupan račun za svaku kupovinu u obliku liste realnih brojeva. Paralelizovati obradu učitanih podataka korišćenjem **dph** paketa. Ne praviti pretpostavku o broju računa u datoteci. Vektorizovan deo koda pisati u modulu Racuni.

   Uputstvo: *Linkovanje vektorizovanog koda Racuni.hs i glavnog programa 5.hs, nakon čega pokrećete izvršni fajl a.out::*
   ghc -fvectorise -package dph-lifted-vseg Racuni.hs
   ghc -package dph-lifted-vseg 5.hs
   ghc -threaded -rtsopts -package dph-lifted-vseg Racuni.o 5.o

Primer 1*

```
Datoteka racuni
  125 10.5 1000
  999 45 12
  74 89 56
Poziv: ./a.out +RTS -N4
Izlaz:
  [1135.5,1056.0,219.0]
```

Primer 2

```
Datoteka racuni
  100 125 10.5 1000
  999 45 12 99 56
Poziv: ./a.out +RTS -N4
Izlaz:
  [1235.5,1211.0]
```

Primer 3

```
Datoteka racuni
  100
  999 21.5
Poziv: ./a.out +RTS -N4
Izlaz:
  [100.0,1020.5]
```