Matematička logika u računarstvu: Drugi praktični kolokvij (Haskell)

PMF MO, ak. god. 2014./2015.

10. lipnja, 2015.

Napomena: Trajanje kolokvija je 120 minuta. Rješenja zadataka pošaljite na ***jan.snajder@fer.hr****. Kolokvij nosi 20 bodova, od kojih je za prolaznu ocjenu potrebno ostvariti barem 10 bodova.*

1. *(3 boda)* Definirajte funkciju `wordHamming w1 w2` koja prebrojava u koliko se znakova razlikuju stringovi `w1` i `w2`.

   ```
   wordHamming "riba" "pita" ⇒ 2
   wordHamming "voda" "vatra" ⇒ 4
   ```

   Definirajte dvije varijante ove funkcije:

   (a) Rekurzivna definicija (`wordHamming1`);

   (b) Definicija pomoću ugrađenih funkcija višega reda (`wordHamming2`).

2. *(3 boda)* Napišite funkciju `indexWords` koja će sve riječi iz stringa zamijeniti jedinstvenim indeksima i vratiti listu indekasa. Iste riječi moraju imati iste indekse. Zanemarite razliku između velikih i malih slova, a sve interpunkcijske znakove tretirajte kao bjeline.

   ```
   indexWords :: String -> [Int]
   indexWords "To be or not to be,that is the question!"
   ⇒ [1,2,3,4,1,2,5,6,7,8]
   ```

   Uputa: Za pretraživanje liste indekasa možete koristiti funkciju `Data.List.lookup`.

3. *(4 boda)* Isprobajte funkciju `group` iz modula `Data.List` koja služi za grupiranje *sortirane* liste u podliste s jednakim elementima:

   ```
   group :: Eq a => [a] -> [[a]]
   group [1,1,2,3,4,4] ⇒ [[1,1],[2],[3],[4,4]]
   ```

   (a) Pomoću funkcije `group` i sažetog zapisa liste napišite funkciju `counts` koja za danu listu vraća listu parova `(element,brojPojavljivanja)`:

   ```
   counts :: Ord a => [a] -> [(a, Int)]
   counts "kikiriki" ⇒ [('i',4),('k',3),('r',1)]
   ```

   (b) Definirajte svoju funkcije za grupiranje elemenata *nesortirane* liste koja će biti tipa:

   ```
   group' :: Eq a => [a] -> [[a]]
   ```

(c) Funkcija `counts` ima tipsko ograničenje koje je strože nego što bi trebalo biti. Bilo bi dobro da možemo prebrojiti elemente za koje je definirana operacija jednakosti (razred `Eq`), ali između kojih nema definiranog poretka (razred `Ord`). Uporabom funkcije `group'` i sažetog zapisa liste definirajte općenitiju inačicu funkcije za prebrojavanje:

```
counts' :: Eq a => [a] -> [(a, Int)]
```

4. *(4 boda)* Definirajte funkciju koja računa umnožak elemenata na neparnim pozicijama u listi:

```
prodOdd :: Num a => [a] -> a
prodOdd [1,4,3,5,7] ⇒ 21
```

Definirajte sljedeće verzije ove funkije:

(a) Rekurzivna izvedba (`prodOdd1`);

(b) Izvedba s akumulatorskim parom (`prodOdd2`);

(c) Izvedba s funkcijom višeg reda `foldr` (`prodOdd3`);

(d) Izvedba s funkcijama `filter` i `product` (`prodOdd4`).

5. *(2 boda)* Definirajte funkciju `partition` koja listu particionira s obzirom na danu predikatnu funkciju.

```
partition :: [a -> Bool] -> [a] -> [[a]]
partition [odd, even, const True] [1..10]
⇒ [[1,3,5,7,9],[2,4,6,8,10],[1,2,3,4,5,6,7,8,9,10]]
partition [isUpper, isDigit] "Catch 22" ⇒ ["C","22"]
partition [] "Whatever" ⇒ []
```

6. *(4 boda)* Napišite funkciju za sortiranje stapanjem (engl. *merge sort*), ali s rekurzijom nad tri liste (*3-way merge sort*), tj. lista se dijeli na tri podliste, a nakon sortiranja podlisti stapaju se istovremeno sve tri podliste.

```
mergeSort3 :: Ord a => [a] -> [a]
```

Funkciju ostvarite pomoću sljedeće dvije funkcije:

(a) Funkcija koja dijeli listu na tri liste podjednake veličine (na koji god način):

```
split3 :: [a] -> ([a], [a], [a])
```

(b) Funkcija koja u pravilnom poretku (uzlazno) stapa elemente triju lista:

```
merge3 :: Ord a => [a] -> [a] -> [a] -> [a]
merge3 [1,3,8,10] [2,4,9] [5,6,7] ⇒ [1,2,3,4,5,6,7,8,9,10]
```

Obratite pažnju na to da pokrijete sve moguće slučajeve ulaznih listi.

(c) (*bonus*) Definirajte funkciju `mergeSortN :: Ord a => Int -> [a] -> [a]` koja listu sortira stapanjem s rekurzijom nad `n` listi.