Matematička logika u računarstvu: Ponovljeni drugi praktični kolokvij (Haskell)

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

23. lipnja, 2015.

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

1. *(3 boda)* Napišite funkciju `vigenere key s` koja string `s` šifrira Vigenèreovom šifrom s ključem (stringom) `k`. Pretpostavite da je ključ sastavljen od malih slova `a-z`. Prije šifriranja poruke, poruku je potrebno pretvoriti u mala slova i iz nje izbaciti sve znakove koji nisu alfabetski, uključivo razmak (koristite funkciju `Data.Char.isAlpha`). Za pretvorbu znaka u ASCII-kôd i obrnuto koristite funkciju `chr` odnosno `ord`. Od koristi vam može biti funkcija `cycle`.

   ```
   vigenere "az" "abba" ⇒ "aabz"
   vigenere "charlie" "it takes a revolution to make a solution"
   ⇒ "katrvmwcyemztyvpoeewqcrerdwpwaify"
   ```

2. *(5 bodova)*

   (a) Napišite rekurzivnu definiciju funkcije `nubRight` za izbacivanje duplikata iz liste. Duplikatom se smatraju svi elementi koji se pojavljuju nadesno od prvog pojavljivanja elementa.

   ```
   nubRight :: Eq a => [a] -> [a]
   nubRight "kikiriki" ⇒ "kir"
   ```

   (b) Napišite rekurzivnu definiciju za `nubLeft`:

   ```
   nubLeft :: Eq a => [a] -> [a]
   nubLeft "kikiriki" ⇒ "rki"
   ```

3. *(4 boda)* Napišite rekurzivnu definiciju funkcije za zbrajanje elemenata na parnim pozicijama u listi:

   ```
   sumEven :: Num a => [a] -> a
   sumEven [1,4,3,5,7] ⇒ 9
   ```

   Definirajte sljedeće verzije ove funkije:

   (a) Rekurzivna izvedba (`sumEven1`);

   (b) Izvedba s akumulatorskim parom (`sumEven2`);

   (c) Izvedba s funkcijama višega reda (`sumEven3`).

   Objasnite vremensku i prostornu složenost ovih funkcija.

4. *(3 boda)* Definirajte `decamel`, funkciju koja identifikator (niz znakova) pretvara iz zapisa *camel case* u zapis s odvojenim riječima. Možete prepostaviti da se nizovi znakova sastoje isključivo od afanumeričkih znakova.

```
decamel :: String -> String
decamel "random" ⇒ "random"
decamel "wordCount" ⇒ "word count"
decamel "AbstractSingletonProxyFactoryBean"
⇒ "abstract singleton proxy factory bean"
decamel "" ⇒ error "identifier is empty"
decamel "contains Whitespace"
⇒ error "input not in camel case format"
```

5. *(5 bodova)* Funkcije u Haskellu ne mogu se ispitivati na jednakost niti se ne može izračunavati njihov inverz. Međutim, takvu provjeru možemo implementirati za funkcije definirane nad konačnim domenama.

   (a) Definirajte funkciju `funEq` jesu li dvije unarne funkcije identične nad zadanom domenom.

```
funEq :: Eq b => [a] -> (a -> b) -> (a -> b) -> Bool
funEq [0..999] succ succ ⇒ True
funEq [0..]  odd even ⇒ False
funEq [0..]  even even ⇒ ⊥
funEq [(1,1),(2,2),(3,3)] fst snd ⇒ True
```

   (b) Definirajte funkciju `tabulate` koja tabulira zadanu funkciju nad zadanom domenom (popisuje parove ulaz-izlaz).

```
tabulate :: [a] -> (a -> b) -> [(a,b)]
tabulate [0..3] odd ⇒ [(0,False),(1,True),(2,False),(3,True)]
tabulate [0..5] succ ⇒ [(0,1),(1,2),(2,3),(3,4),(4,5),(5,6)]
```

   (c) Definirajte funkciju koja provjerava je li zadana funkcija, restringirana nad danom domenom, injekcija.

```
injective :: Eq b => [a] -> (a -> b) -> Bool
injective [0..5] succ ⇒ True
injective [0..5] odd ⇒ False
```

   (d) *(bonus)* Definirajte funkciju `inv xs f` koja računa inverz funkcije `f` nad domenom `xs`. Ako je `f` neinjektivna, onda nema inverza, no u tom slučaju vratite funkciju koja preslikava na jedan (proizvoljan) element iz skupa mogućih inverznih vrijednosti. Ako inverzna funkcija nije definirana za zadani element, funkcija treba prekinuti s greškom.

```
inv :: Eq b => [a] -> (a -> b) -> (b -> a)
invOdd = inv [0..9] odd
invSucc = inv [0..9] succ
invOdd True ⇒ 1
invSucc 5 ⇒ 4
invSucc 0 ⇒ error "no image"
funEq [1..10] invSucc pred ⇒ True
funEq [0..10] invSucc pred ⇒ error "no image"
```