Napredno funkcionalno programiranje

Zadaća 1

Amer Hasanović

Problem 1

Izvod proizvoljne funkcije f definira se sljedećom relacijom:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

a može se približno izračunati numerički:

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

gdje je Δx mala konstanta.

Implementirajte `Haskell` funkciju `izvod` koja vraća izvod proizvoljne funkcije jedne promjenjive. Primjer upotrebe funkcije:

```
ghci> :load P1.hs
[1 of 1] Compiling Main             ( P1.hs, interpreted )
Ok, one module loaded.
ghci> f x = x*sin(x) + x^2
ghci> g = izvod f
ghci> g 5
10.463067868357712
```

Za konstantu Δx koristiti vrijednost `0.001`.

Problem 2

Potrebno je implementirati funkciju `sumaPovrsina`, za računanje sume površina geomtrijskih oblika koji su dati u `Haskell` listi, pri čemu su validni oblici pravougaonici i krugovi.

Potrebno je implementirati i sve neophodne popratne elemente koda, kako bi se funkcija mogla koristiti kao u sljedećem primjeru:

```
ghci> :load P2.hs
[1 of 1] Compiling Main             ( P2.hs, interpreted )
Ok, one module loaded.
ghci> sum
sum           sumaPovrsina
ghci> sumaPovrsina [Krug 2, Krug 3, Pravougaonik 2 4]
48.82
ghci>
```

Problem 3

Implementirajte funkcije `fr` i `fl` koje imaju sljedeće potpise:

```
fr :: (a -> b -> b) -> b -> [a] -> b
fl :: (b -> a -> b) -> b -> [a] -> b
```

Kada se pozovu odgovrajućim argumentima funkcije trebaju da proizvode rezultat aplicirajući funkciju, koja je prvi argument u oba slučaju, na ostalim argumentima na sljedeći način:

```
fr f x [1..10]
f 1 (f 2 (f 3 (f 4 (f 5 (f 6 (f 7 (f 8 (f 9 (f 10 x)))))))))
fl f x [1..10]
f (f (f (f (f (f (f (f (f x 1) 2) 3) 4) 5) 6) 7) 8) 9) 10
```

Koristeći funkciju `fl` implementirajte funkciju `myReverse` koja uzima listu popunjenu elementima proizvoljnog tipa, a vraća listu sa elementima u obrnutom redosljedu.

Primjer upotrebe:

```
ghci> myReverse [4,3,1,2]
[2,1,3,4]
```

Problem 4

Potrebno je implementirati funkciju `verificiraj` koja uzima broj kreditne kartice, a vraća rezultat provjere validnosti broja, i to na sljedeći način:

1. Broj se konvertuje u listu cifri. npr `164` postaje `[1, 6, 4]`
2. Svaka druga cifra u listi, od pozada, se udupla. npr `[1, 6, 4]` postaje `[4, 12, 1]`
3. Sumirati sve cifre u listi, tretirajući cifre u eventualno višecifrenim brojevima separatno. npr `[4, 12, 1]` postaje `4 + 1 + 2 + 1`
4. Ukoliko je ostatak dijeljenja dobivene sume sa `10` jednak `0`, broj je validan, u suprotnom, ne.

Primjer upotrebe funkcije:

```
ghci> :load P4.hs
[1 of 1] Compiling Main             ( P4.hs, interpreted )
Ok, one module loaded.
ghci> verificiraj  5594589764218858
True
ghci> verificiraj  1234567898765432
False
```

Problem je neophodno riješiti razbijajući ga na manje dijelove implementacijom sljedećih funkcija:

```
-- zadnjaCifra vraća zadnju cifru u broju
zadnjaCifra :: Integer -> Integer

-- izbaciZadnjuCifru eliminira zadnju cifru iz broja
izbaciZadnjuCifru :: Integer -> Integer

-- uCifre konvertuje broj u listu njegovih cifri
-- na način da se na početku liste nalazi zadnja cifra,
-- nakon nje predzadnja, itd ...
uCifre :: Integer -> [Integer]

-- duplirajSvakiDrugi duplira svaki drugi element liste
duplirajSvakiDrugi :: [Integer] -> [Integer]

-- sumirajSveCifre implementira treći korak u algoritmu
sumirajSveCifre :: [Integer] -> Integer
```

Funkcija `verificiraj` je adekvatna kompozicija gore navedenih funkcija.

Pokušati što više koristiti funkcije iz `Prelude`-a koje rade na listama.