Csoportok

6. házi feladat

Kategória:
Házi feladatok
Végső határidő:
3/31/2025, 11:59 PM (Beadva)
Próbálkozások száma:
Korlátlan
Kiírta:
Erdei Zsófia

Leírás:

Rekurzió, elágazás

Most már vannak elágazások, így van néhány felhívás a használhatukkal kapcsolatban:

- Ha egy függvény eredménye Bool típusú kell legyen, akkor az elágazás teljesen felesleges dolog, így nem is lehet használni azokban a függvényekben.
- Egy Bool típusú értéket egyenlőségvizsgálni megint teljesen felesleges (pl. (1 > 2) == True, már az 1 > 2 megválaszolja ugyanazt), így szintén nem szabad úgy használni, csúnya dolog, redundáns, attól csak feleslegesen több a kód, nem jobb.
- if-then-else továbbra sem használható! Tessék megszokni, hogy vannak guard-ok.

Más is van, amit eddig talán nem tisztáztam (listás dolgok):

- Haskellben a végtelen lista az alapértelmezett feltevés, ettől csak akkor lehet eltérni, ha a feladat kifejezetten kiemeli, hogy csak véges listára lehet/kell/fog működni az adott függvény. Ez azt jelenti, hogy amely függvényeknek végtelen listára is kell működniük, azokban TILOS length, sum, product és más hasonlóan végtelenre nem működő függvényeket használni.

0. Modul

Hozz létre egy modult `Hazi6` néven!

1. Karakter-átalakítás

Alakítsd át egy szöveg 3. karakterét nagy betűre definiálva a `toUpperThird` nevű függvényt. Minden más karakter maradjon ugyanaz. Ha nincs 3. karakter, akkor adja vissza a kapott szöveget (hiszen nincs mit átalakítani). Ha a 3. helyen szereplő karakter már nagy betű, akkor is maradjon ugyanaz a visszaadott szöveg. Használj mintaillesztést! Ne használj egyenlőségvizsgálatot vagy elágazást vagy indexelést, se `take` , se `drop` függvényt, se rekurziót!

Segítség: A `Data.Char` modulnak a `toUpper` függvénye jól jöhet.

```
toUpperThird :: String -> String
```

2. Rendezett-e

Definiáld az `isSorted` nevű függvényt, amely ellenőrzi rendezhető elemek listájáról, hogy az elemei növekvő sorrendben vannak-e. Ha a lista növekvően rendezett, akkor végtelen lista esetén a függvény nem terminál (remélhetőleg értelemszerű okok miatt).

```
isSorted :: Ord a => [a] -> Bool
```

3. Titkok tudója

Definiáld a `cipher` függvényt, amely egy titkosított szövegből kinyeri az első olyan kettő hosszú karaktersort, amelyet számjegy követ. Ha nincs ilyen, akkor az eredmény legyen üres `String` . A megoldásban használj mintaillesztést és rekurziót! A `take` , `drop` , `length` és általánosabb variánsaik teljesen feleslegesek, épp ezért nem használhatóak!

Segítség: Az egyes karakterek azonosításához használjuk a `Data.Char` függvényeit. Hogy melyiket? Azt kell nektek megtalálni.

4. Dupla elemek

Definiáld a `doubleElements` függvényt, amely minden egyes lista elemét egyenként megkettőzi egymás után! Használj rekurziót!

```
doubleElements :: [a] -> [a]
```

5. Sok szóköz

Definiáld a `deleteDuplicateSpaces` függvényt, amely egy szövegből eltávolítja a több egymás mellett álló szóközöket, azokból egyet meghagyva. A szöveg eleji és szöveg legvégi szóközöket teljes egészében dobja el a függvény. Feltehető, hogy a szöveg kisbetűből, nagybetűből, számból és szóközből áll csak.

```
deleteDuplicateSpaces :: String -> String
```

Segítség: Ne bonyolítsuk túl a megoldást!

Tesztesetek

```
toUpperThird "a" == "a"
toUpperThird "az" == "az"
toUpperThird "kap" == "kaP"
toUpperThird "alma" == "alMa"
toUpperThird "ALMA" == "ALMA"
take 20 (toUpperThird (repeat 'b')) == "bbBbbbbbbbbbbbbbbbb"
isSorted ([] :: [Integer])
isSorted [1::Int]
isSorted [1::Integer]
isSorted [1::Double]
isSorted "a"
isSorted [5,6,9,10]
isSorted "aabcddddefg"
isSorted [(-2),(-1),1,9,10,19]
isSorted "adn"
not (isSorted [10,9,8,7,6,5,4,3,2,1,0])
not (isSorted "alma")
not (isSorted [10,9..])
not (isSorted ([1..10] ++ [9,8..]))
cipher "PYdg7iT4vdO0n4AgmGfUpRzogAf" == "dg"
cipher "PYdgaiTLvdOKnAAgmGfUpRzogA4" == "gA"
cipher "4vkYyAO174midQTt0" == "AO"
cipher "BwxwEwqCKHuMTAaPn" == ""
cipher ['\0'..] == "./"
cipher "dM7" == "dM"
cipher "777" == "77"
cipher "Kmz" == ""
cipher "Zk" == ""
cipher "T4" == ""
cipher "" == ""
doubleElements [1,2,3] == [1,1,2,2,3,3]
null (doubleElements [])
doubleElements "alma" == "aallmmaa"
take 10 (doubleElements [0..]) == [0,0,1,1,2,2,3,3,4,4]
deleteDuplicateSpaces "alma  szilva    barack" == "alma szilva barack"
deleteDuplicateSpaces "    alma  szilva    barack    eper    " == "alma szilva barack eper"
deleteDuplicateSpaces "    alma  szilva    barack    eper" == "alma szilva barack eper"
deleteDuplicateSpaces "    alma  szilva    barack    eper " == "alma szilva barack eper"
deleteDuplicateSpaces "alma  szilva    barack    eper  " == "alma szilva barack eper"
deleteDuplicateSpaces " alma  szilva    barack    eper    " == "alma szilva barack eper"
take 12 (deleteDuplicateSpaces (cycle "a b")) == "a ba ba ba b"
null (deleteDuplicateSpaces "       ")
null (deleteDuplicateSpaces "")
```

Git tároló

Útvonal:

`https://tms.inf.elte.hu/git/8256/vl2l98/w7a163e3ea0c3ae10864550e79`

Használat:

`git clone https://tms.inf.elte.hu/git/8256/vl2l98/w7a163e3ea0c3ae10864550e79`

Megoldás

Név:
solution.zip
Feltöltés ideje:
3/26/2025, 11:22 AM

Értékelés:
Státusz:
Elfogadva
Feltöltések száma:
2
Értékelte:
Szávó Tamás
Megjegyzések:
"Előző megoldásom | y >= x = isSorted xs volt, arra is lefutott sikeresen a tesztelő, viszont hibás volt a függvény pl.: isSorted [1,2,1,2,5,6,7,8,9,10] == True"
Érdekes, pont ilyen teszt nem volt. Elég szerencsés hiba volt, de biztos ami biztos most már raktam bele ilyen tesztet is, köszi, hogy szóltál
A megoldás egyébként jó

Automatikus tesztelés eredményei

#1