Csoportok

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 12

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 91

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 92

6. házi feladat

Kategória: Házi feladatok
Elérhető: 2022. 10. 21. 23:59
Pótolható határidő: 2022. 11. 04. 23:59
Végső határidő: 2022. 11. 06. 23:59
Kiírta: Erdei Zsófia

Leírás:

Házi feladat

A házi feladatot egy Homework6 nevű modulként kell beadni. Minden definiálandó függvényhez adjuk meg a hozzá tartozó típusszignatúrát is!

Fontos: a feltöltött tömörített zip állományban egyetlen .hs kiterjesztésű fájl szerepeljen (*hazi.txt*, *Homework.hs.txt* nem felel meg)!

Tipp: Ha szükség van rá, a feladatokat bontsd fel részfeladatokra és definiáljatok segédfüggvényeket!

A bónusz feladat megoldása nem kötelező (és azt nem teszteli az automatikus tesztelő).

Listák különbsége

Írjuk meg a saját listakülönbség függvényünket! Működése legyen hasonló a halmazelméletből ismert különbség műveletére, azaz azokat az elemeket tartalmazza a különbséglista, amely az első listának az eleme, de a másodiknak nem! (A `Data.List.(\\)` függvény nem használható!)

```
listDiff :: Eq a => [a] -> [a] -> [a]
```

Az alábbi tesztesetek közül mindegyiknek `True` -t kell adnia:

```
listDiff "Haskell" "the best" == "Hakll"
listDiff "Cannot predict" "the future" == "Cannopdic"
listDiff "Be the type of person" "you want to meet" == "Bhpfprs"
listDiff "Eotvos Lorand" "Tudomanyegyetem" == "Evs Lr"
```

Szólánc játék

Adjuk meg, hogy helyesen játszották-e a szólánc játékot! A játék célja, az előzőleg elhangzó szó utolsó betűjével kell egy újabb szót alkotni. A szavakat egy-egy szóközzel választottuk el. A feladat megoldásában segítséget nyújthat a words függvény.

Az alábbi tesztesetek közül mindegyiknek `True` -t kell adnia:

```
validGame "alma asztal lo olalkodik kutya" == True
validGame "konverv veg golya abrosz zsifaf fules" == True
validGame "erdo osztokel nyugat tabortuz" == False
validGame "zokog guzsalyas sararany nyul leng" == False
```

Egyelemű listák

Számold meg, hogy hány egyelemű lista szerepel a listában!

```
countSingletons :: [[a]] -> Int
```

Az alábbi tesztesetek közül mindegyiknek `True` -t kell adnia:

```
countSingletons [[1..], [2], [3,4], [5], []] == 2
countSingletons [] == 0
countSingletons [[]] == 0
countSingletons [[1..1], [1..2], [1..3], [2..], [3],
[5], [10], [], [10..]] == 4
```

Paritás ellenőrzése

Döntsd el, hogy a lista elemei mind ugyanazt a maradékot adják-e kettővel osztva.

```
sameParity :: [Int] -> Bool
```

Az alábbi tesztesetek közül mindegyiknek `True` -t kell adnia:

```
sameParity []              == True
sameParity [3]             == True
sameParity [1..10]         == False
sameParity [1,3..10]       == True
sameParity [2, 42, 0, 8]   == True
sameParity (1:[2, 42, 0, 8])  == False
```

Leghosszabb egyező karakterlánc

A `longestChain` nézze meg, hogy milyen hosszú az a leghosszabb karaktersorozat egy szövegen belül, ami azonos karakterekből áll!

```
longestChain :: String -> Int
```

Az alábbi tesztesetek közül mindegyiknek `True` -t kell adnia:

```
longestChain "2111234" == 3
longestChain "0023212212222" == 4
longestChain "23232323232" == 1
longestChain "+++++!!!-------" == 7
longestChain "++!++-+!!-!---||-" == 3
longestChain "(>.<),--(o.o)" == 2
longestChain "0" == 1
longestChain "" == 0
```

Szöveg normalizálása

Definiáljátok azt a függvényt, amely egy szöveget normalizál a következőképpen: az angol ábécé kisbetűit nagybeűkké alakítja, az angol ábécé nagybetűit változatlanul hagyja, és minden egyéb karaktert (írásjeleket, whitespace-eket, stb.) eltávolít a szövegből.

```
normalizeText :: String -> String
```

Az alábbi tesztesetek közül mindegyiknek `True` -t kell adnia:

```
normalizeText "sos"             == "SOS"
normalizeText ".! 7"            == ""
normalizeText "Save our souls!" == "SAVEOURSOULS"
normalizeText "limonádé"        == "LIMOND"
```

Bónusz (nem kötelező)

Definíció

Szeretnénk egységes, egyértelmű nevet adni ezeknek a hozzárendeléseknek, ezért ebben az egy esetben tekintsünk el a formális matematikai definíciótól és minden ilyen leképezést/hozzárendelést/függvényt nevezzünk `Function` -nek. Többféle módon

megadhatunk ilyen függvényeket, most tegyük ezt meg egy párok listájaként, amely leírja, mely bemenethez, mely kimenet tartozik. Vagyis `[(1,"alma"), (5,"barack")]` egy olyan függvény, amely az `1` -es értékhez az `"alma"` , míg az `5` -ös értékhez a `"barack"` szavakat rendeli. Vezessük be az alábbi típusszinonímát:

```
type Function a b = [(a, b)]
```

Tranzitív

Definiáljuk a `isTransitive` függvényt, amely vár egy `Function` -t és eldönti tranzitív-e. Tranzitív egy reláció akkor, ha `∀ a, b, c ∈ A` : ha `a` relációban áll `b` -vel és `b` relációban áll `c` -vel, akkor `a` relációban áll `c` -vel

```
isTransitive :: Eq a => Function a a -> Bool
```

A következő teszteseteket a terminálba másolva mindre igazat kell kapni

A tesztesetek futtatásához a **Data.List** modul importálása szükséges!

Tesztesetek

```
isTransitive [] == True
isTransitive [(1,2)] == True
isTransitive [(1,2), (2,5)] == False
isTransitive [(1,2), (2,5), (1,5)] == True
isTransitive [(1,1),(1,2),(1,3),(1,4),(1,5),(2,2),(2,3),(2,4),(2,5),(3,3),(3,4),(3
and [ not (isTransitive (tail x)) | x <- permutations [(z,y) | z <- [1..3], y <- [
```

Megoldás

Letöltés

Név: Homework6.zip
Feltöltés ideje: 2022. 11. 06. 23:08
Értékelés:
Státusz: Elfogadva
Feltöltések száma: 1
Értékelte: Erdei Zsófia
Megjegyzések:

Automatikus tesztelés eredményei

```
A megoldás átment a teszteken
```