Házi feladat

A házi feladatot egy Homework1 nevű modulként kell beadni. Minden definiálandó függvényhez adjuk meg a hozzá tartozó típusszignatúrát is! A feladatok után a zárójelben lévő név azt jelzi, milyen néven kell definiálni az adott függvényt, kifejezést. A forrásfáljban ügyeljetek arra, hogy minden kifejezés rendelkezzen helyes típusszignatúrával!

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

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.

A megvalósításhoz használjuk a `map` és `filter` függvényeket!

```
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"
```