Előzetes tudnivalók

Használható segédanyagok:

- Haskell könyvtárak dokumentációja,
- Hoogle,
- a tárgy honlapja, és a
- Haskell szintaxis összefoglaló.

Más segédeszköz nem használható.

Ha bármilyen kérdés, észrevétel felmerül, azt a gyakorlatvezetőnek kell jelezni, **nem** a diáktársaknak!

A feladatsor megoldására 20 perc áll rendelkezésre (+ 5 perc feltöltésre)

A feladatok tetszőleges sorrendben megoldhatóak. A pontozás szabályai a következők:

- Minden teszten átmenő, a feladat kikötéseinek megfelelő megoldás érhet teljes pontszámot.
- Funkcionálisan hibás (valamelyik teszteseten megbukó) megoldás nem ér pontot.
- Fordítási hibás kód esetén a teljes zh 0 pontos.

Ha hiányos/hibás részek lennének a feltöltött megoldásban, azok kommentben szerepeljenek.

Tekintve, hogy a tesztesetek, bár odafigyelés mellett íródnak, nem fedik le minden esetben a függvény teljes működését, határozottan javasolt még külön próbálgatni a megoldásokat beadás előtt!

Az elméleti kérdésekre adott válaszokat a forráskódban kell elhelyezni, kommentben. Minden függvénynek meg kell adni a típusszignatúráját is. A függvények elvárt neve és típusa meg van adva. `Zarthelyi2` néven kell deklarálni a modult. A `.hs` fájlt `.zip`-be tömörítve kell beadni.

Elméleti kérdések (1 pont / kérdés)

1. Mi a különbség `Int` és `Integer` típusok között? (1 pont)
2. Mit jelent az, hogy a Haskell nyelv lusta kiértékelésű? Fogalmazd meg röviden, illetve írj egy példát, hol lehet ezt kihasználni! (1 pont)

Gyakorlati feladatok

Fűzögetett párok (1 pont)

Definiáljuk a `concatPairs` függvényt, amely a párokban kapott listákat összefűzi.

```
concatPairs :: [([a],[a])] -> [[a]]



concatPairs [([1,2],[3,4])] == [[1,2,3,4]]
concatPairs [([1,2],[3,4]), ([5,6],[7,8])] == [[1,2,3,4], [5,6,7,8]]
concatPairs [] == ([] :: [[Int]])
```

```
concatPairs (replicate 100 ([1],[2])) == replicate 100 [1,2]
take 150 (concatPairs (repeat ([1],[2]))) == replicate 150 [1,2]
```

Forduljanak a hármasok (1 pont)

Definiáljuk a `reverseTriples` függvényt, ami egy 3-as tuple-ökből álló listában a rendezett 3-asok komponenseinek sorrendjét megfordítja. A megoldásban használj mintaillesztést és rekurziót! Ne használj listagenerátort!

```
reverseTriples :: [(a,b,c)] -> [(c,b,a)]
```

```
reverseTriples [(1,2,3), (3,2,1)] == [(3,2,1), (1,2,3)]
reverseTriples [(1, 'a', True), (4, 'X', False)] == [(True, 'a', 1), (False, 'X', 4)]
take 2 (reverseTriples (cycle [("alma", 15, 'k')])) == [('k', 15, "alma"), ('k', 15, "alma")]
```

A Fekete-tó viharjai (2 pont)

A roxforti birtokon elhelyezkedő Fekete-tavon kiépítettek egy viharjelző rendszert. A méréseket 3 komponensű tuple-ök reprezentálják: `(22.3, 'N', True)`, az első két komponens rendre a mért csapadék mennyiségét és a szél irányát adja meg. Az utolsó komponens, azt a tényt rögzíti, hogy a vihar közönséges volt-e vagy mágikus (ha **igaz**, akkor volt **mágikus**).

Definiáljuk a `stormInHogwarts` függvényt, amely kap egy ilyen 3 komponensű tuple-ökből álló listát, és megmondja, hogy összesen mennyi csapadék esett mágikus viharokban. A listáról feltehető, hogy véges!

```
stormInHogwarts :: [(Double, Char, Bool)] -> Double
```

```
stormInHogwarts [(22.3, 'S', False), (2.4, 'N', False), (5.1, 'W', False)] == 0
stormInHogwarts [(22.3, 'S', False), (2.4, 'N', True), (5.1, 'W', False)] == 2.4
stormInHogwarts [(22.3, 'S', False), (2.4, 'N', True), (0, 'W', True)] == 2.4
stormInHogwarts [(22.3, 'S', False), (2.4, 'N', True), (0, 'W', True)] == 2.4
stormInHogwarts [(22.3, 'S', False), (2.4, 'N', True), (6.7, 'W', True)] == 9.1
```