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 35 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. `Zarthelyi2Pot` néven kell deklarálni a modult. A `.hs` fájlt `.zip`-be tömörítve kell beadni.

Elméleti kérdések (1 pont / kérdés)

1. Mit jelent a homogén adatszerkezet? Fogalmazd meg, és írj rá egy példát is! (1 pont)
2. Mi a különbség az `init` és `tail` között? Fogalmazd meg, és írj rá egy példát is! (1 pont)

Gyakorlati feladatok

Párok kibontása (1 pont)

Definiáljuk a `unzipPairs` függvényt, amely a párokat tartalmazó listában kibontja a párokat! A sorrend maradjon meg! **Ne használj magasabb rendű függvényt!**

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



null (unzipPairs [])
unzipPairs [(1,2),(3,4)] == [1,2,3,4]
unzipPairs [(10,20),(30,40),(50,60)] == [10,20,30,40,50,60]
```

```
unzipPairs (replicate 100 ("ez","az")) == take 200 (cycle ["ez","az"])
take 100 (unzipPairs (repeat (2,3))) == take 100 (cycle [2,3])
```

Kicserélés (1 pont)

Definiáljuk a `replace` függvényt, ami egy szövegben lecseréli a paraméterként kapott karakter összes előfordulását egy szintén paraméterként kapott karakterre. **Ne használj magasabb rendű függvényt!**

```
replace :: Char -> Char -> String -> String
```

```
replace 'a' 'e' "" == ""
replace 'a' 'e' "almafa" == "elmefe"
replace ' ' '_' "ez egy szoveg" == "ez_egy_szoveg"
replace '@' 'a' "kuk@c" == "kukac"
replace 'x' 'y' "haskell" == "haskell"
(replace 't' 'g' . replace 'c' 'd' . replace 'a' 'o' $ "cat") == "dog"
take 31 (replace 'p' 'c' (cycle "parrot")) == "carrotcarrotcarrotcarrotcarrotc"
```

A rövidebb lista hossza (2 pont)

Definiáljuk a `lengthOfShorter :: Num i => [a] -> [b] -> i` függvényt, amely a két paraméterként kapott lista közül a visszaadja a rövidebb lista hosszát. **Ne használj magasabb rendű függvényt!**

```
lengthOfShorter [] [] == 0
lengthOfShorter [] [1..] == 0
lengthOfShorter [1..] [] == 0
lengthOfShorter [1..5] [1..10] == 5
lengthOfShorter [1..10] [1..5] == 5
lengthOfShorter ['a'..'j'] [1..] == 10
lengthOfShorter [30..] [id,id,id,id,id,id] == 6
```