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 25 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. `Zarthelyi3` néven kell deklarálni a modult. A `.hs` fájlt `.zip`-be tömörítve kell beadni.

Elméleti kérdés (1 pont)

- Mit nevezünk magasabb rendű függvénynek?

Gyakorlati feladatok

Függvénnyé alakítás (2 pont)

Definiálj egy függvényt, amely azonos típusú elemeket tartalmazó rendezett párok listájából definiál egy listányi függvényt, amely egy `Bool` típusú értéket kap paraméterül, amely szerint eldönti, hogy a pár első eleme (`True` érték esetén) vagy a második eleme (`False` érték esetén) lesz az eredmény.

```
functionGen :: [(a,a)] -> [Bool -> a]
```

```
null (functionGen [])
(functionGen [(1,2),(3,4),(9,0),(12,-1)] !! 2) True == 9
(functionGen [(1,2),(3,4),(9,0),(12,-1)] !! 2) False == 0
```

```
[(f False, f True) | f <- functionGen [('a','b'),('z','d')]] == [('b','a'),('d','z')]
take 20 [(g 0, h 2) | (g,h) <- [(f True, f False) | f <- functionGen [((+i),(*i)) | i <- [1..]]]] == zip
```

Adott tulajdonságú elemek törlései (3 pont)

Definiálj egy függvényt, amely adott tulajdonságú *egy* elemet kitöröl egy listából az összes lehetséges módon. Tehát az összes lehetséges módon törölj az összes adott tulajdonságú elemek közül pontosan egyet.

```
deletionsBy :: (a -> Bool) -> [a] -> [[a]]
```

A tesztek futtatásához szükség van a `Data.Char` modulra.

```
deletionsBy (not . null) [[1,2],[],[],[3,4],[5]] == [[[],[],[3,4],[5]],[[1,2],[],[],[5]],[[1,2],[],[],[3,
deletionsBy even [1,2,3,4,5,6] == [[1,3,4,5,6],[1,2,3,5,6],[1,2,3,4,5]]
deletionsBy isUpper "SzIlVa" == ["zIlVa","SzlVa","SzIla"]
null (deletionsBy (const False) [])
null (deletionsBy (const False) [1,2,3])
deletionsBy (const True) [1,2,3] == [[2,3],[1,3],[1,2]]
take 22 ((deletionsBy odd [1..]) !! 10) == [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,22,23]
take 12 ((deletionsBy odd [1..]) !! 5) == [1,2,3,4,5,6,7,8,9,10,12,13]
```