A házi feladatot egy `Homework4` nevű modulként kell beadni. Figyeljetek arra, hogy a függvényeitek a module szóval egy "oszlopba" kerüljenek, azaz ne legyenek beljebb húzva! Minden definiálandó függvényhez adjuk meg a hozzá tartozó típus szignatúrát is! (Ezt most megadtam, a saját modulotokba is másoljátok be a definíciótok elé.)

A feladatokat kötelező rekurzióval megoldani és minden függvény működjön végtelen listákra is hacsak a feladat nem kéri!

- Definiáljuk a `concatList` függvényt amely egy listák listáját összefűz egy listába! (`concatList :: [[a]] -> [a]`)
- Definiáljuk a `factorial` függvényt amely egy szám faktoriálisát számolja ki! Negatív szám esetén tegyünk úgy mintha `0`-t kaptunk volna paraméterül. (`factorial :: Int -> Int`)
- Definiáljuk az `index` függvényt ami egy lista `n`-edik elemét adja vissza. `n < 0` vagy `n >= length xs` esetén a függvénynek nem kell működnie (Ilyenkor használhatjuk az `error :: String -> a` függvényt, hogy fatal exceptiont adjunk vissza). (`index :: Int -> [a] -> a`)
- Definiáljuk a `zipLists` függvényt amely két lista elemeit párhuzamosan összefűzi! Ha valamelyik listából elfogy az elem, hagyjuk abba a rekurziót! (`zipLists :: [a] -> [b] -> [(a,b)]`)
- Definiáljuk a `fakeDropSort` függvényt amely kiszedi azokat az elemeket a listából amelyek nem kisebbek mint az azt követő elem a listában! (`fakeDropSort :: Ord a => [a] -> [a]`)
- Definiáljuk az `onlyIncreasing` függvényt, amely egy listából leszűri azon tupleöket amelyekben nem növekvő sorrendben vannak az elemek! (`onlyIncreasing :: Ord a => [(a,a,a)] -> [(a,a,a)]`)

Tesztek a működésre:

```
zipLists [] [1..] == []
zipLists [1..] [] == []
zipLists [1,2,3] [4,5,6,7] == [(1,4), (2, 5), (3, 6)]
concatList [[]] == []
concatList [[1,2,4], [], [5]] == [1,2,4,5]
concatList [[], [], [], [1,2,3]] == [1,2,3]
take 10 (concatList [[1..10], [1..]]) == [1..10]
onlyIncreasing [(1,2,3), (4,5,6)] == [(1,2,3), (4,5,6)]
onlyIncreasing [(4,5,6), (1,2,3)] == [(4,5,6), (1,2,3)]
onlyIncreasing [(1,2,2), (6,5,7)] == []
factorial 0 == 1
factorial 2 == 2
factorial 10 == 3628800
index 0 [1,2,3] == 1
index 1 [1,2,3] == 2
index 3 [1,2,3,5] == 5
index 0 [10..] == 10
fakeDropSort [1,2,3,4] == [1,2,3,4]
fakeDropSort [2,2,3,5,4] == [2,3,4]
fakeDropSort [2,3,4,2,5] == [2,3,2,5]
take 10 (fakeDropSort [1..]) == [1..10]
factorial (-1) == 1
index 1 [10..] == 11
```