Házi feladat

A házi feladatot egy `Homework6` nevű modulként kell beadni. Minden definiálandó függvényhez adjuk meg a hozzá tartozó típus szignatúrát is! A feladatok után a zárójelben lévő név azt jelzi, milyen néven kell definiálni az adott függvényt, kifejezést. A feladatok megoldása során törekedjetek arra, hogy a gyakorlaton tanult módszereket, megoldási meneteket használjátok.

Van-e igaz elem

Implementáld a `some :: [Bool] -> Bool` függvényt, amely eldönti feltételeket tartalmazó listáról, hogy van-e igaz eleme. Ebben a feladatban az or függvény használata nem megengedett.

Teszt:

```
some [succ 'a' /= 'b', 1 + 1 /= 2, 10 < 20] == True
some [succ 'a' /= 'b', 1 + 1 /= 2, 10 >= 20] == False
some [] == False
```

Részlista

Implementáld a `sublist :: Int -> Int -> [b] -> [b]` függvényt, ami kivág egy listából egy részt! Az első paraméter az index, ahonnan a részlista kezdődik és a második paraméter a vágás hossza.

```
sublist 2 2 "hello" == "ll"
sublist 0 2 [1..10] == [1,2]
sublist 2 2 [] == []
sublist 8 10 [1..10] == [9,10]
```

Jelszó

Implementáld a `password :: [Char] -> [Char]` függvényt, ami egy karakterlánc minden karakterét a * karakterre cseréli ki.

Teszt:

```
password "akacfa2" == "*******"
password "hunter1234" == "**********"
password ['a'] == ['*']
password [] == []
```

Szótár

Implementáld a `lookup' :: Eq a => a -> [(a, b)] -> b` függvényt, ami kulcs-érték párok listájából kikeresi azt az értéket, ami egy adott kulcshoz tartozik. (A kulcs érték párokat egyszerű tuple-ökkel definiáljuk.)

Teszt:

```
lookup' 2 [(3,"xy"),(2,"abc"),(4,"qwe")] == "abc"
lookup' 42 [(1,2),(3,4),(42,42)] == 42
```

Bináris számok

Implementáld a `toBin :: Integer -> [Int]` függvényt, amely visszaadja egy nemnegatív egész szám kettes számrendszerbeli számjegyeit fordított sorrendben!

Teszt:

```
toBin 0 ==[]
toBin 1 == [1]
toBin 2 == [0,1]
toBin 10 == [0, 1, 0, 1]
```