# HÁZI FELADAT 3.

A házi feladatot egy Homework3 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.)

**FONTOS! A fájlban a `module` definíció előtt egy sorban, kommentben írjátok oda neveteket, Neptun kódotokat!**

## 1. EGYSZERŰ MINTAILLESZTÉS

Adjuk meg azt a függvényt, amely eldönti egy számról, hogy az egy "kis" prím-e! Kis prím alatt az egyjegyű prímeket értjük (2,3,5,7). Ezekre a számokra igazat, minden egyéb számra hamisat adjon vissza! ( `isSmallPrime :: Int -> Bool` )

## 2. EGYSZERŰ MINTAILLESZTÉS

A következő két feladathoz az alábbi linkeken megtaláljátok a műveletek igazságtábláit: implikáció, ekvivalencia. Ezek alapján könnyedén megoldható a feladat.

Adjuk meg azt a függvényt, amely eldönti két logikai változóról, hogy azok **ekvivalensek-e**! ( `equivalent :: Bool -> Bool -> Bool` )

## 3. EGYSZERŰ MINTAILLESZTÉS

Adjuk meg azt a függvényt, amely eldönti két logikai változóról, hogy az első **implikálja-e** a másodikat! ( `implies :: Bool -> Bool -> Bool` )

## 4. KOORDINÁTA-RENDSZER.

Reprezentáljuk a Descartes-féle koordináta-rendszer pontjait rendezett párokkal! A tuple első komponense legyen az x koordináta, a második az y koordináta.

Adjuk meg azt a függvényt, amely középpontosan tükröz egy pontot az origo-ra! ( `invert0 :: (Int, Int) -> (Int, Int)` )

## 5. RACIONÁLIS SZÁMOK I.

Reprezentáljuk a racionális számokat rendezett párokkal! A pár első komponense legyen a számláló, a második pedig a nevező. A műveletek elvégzése után nem feltétlenül muszáj a legegyszerűbb formában megadni az eredményt (tehát nem kell egyszerűsíteni a törtet - bővíteni se bővítsük). Továbbá feltehetjük, hogy egyik szám nevezője sem nulla.

Definiáljuk a racionális összeadást! ( `add :: (Int, Int) -> (Int, Int) -> (Int, Int)` )

## 6. RACIONÁLIS SZÁMOK II.

Definiáljuk a racionális szorzást! ( `multiply :: (Int, Int) -> (Int, Int) -> (Int, Int)` )

## 7. RACIONÁLIS SZÁMOK III.

Definiáljuk a racionális osztást! Feltehetjük, hogy egyik szám számlálója sem nulla. (`divide :: (Int, Int) -> (Int, Int) -> (Int, Int)`)

## 7. LISTA KONSTRUKCIÓ I

Adjuk meg azt a függvényt, amely a kapott paraméterét belerakja egy listába! A végeredmény egy egy elemű lista legyen, amelyben a kapott paraméter szerepel.

```
putIntoList :: a -> [a]
```

## 8. LISTA KONSTRUKCIÓ II

Adjuk meg azt a függvényt, amely két egész számot kap, és előállítja a köztük értelmezett intervallumot! Például `interval 2 5 == [2,3,4,5]`. Abban az esetben, ha az első argumentum nagyobb, mint a második, akkor üres listát adjon vissza!

```
interval :: Int -> Int -> [Int]
```

Segítség: a dokumentum végén.

## B1 - BÓNUSZ 1

Adjuk meg azt a függvényt, amely egy listából kiválogatja a páros számokat!

```
evens :: [Int] -> [Int]
```

### Segítségek

8: Használd a pontpont operátort.