0. Saját modul

Hozz létre egy modult `Hazi9` néven!

1. Óra típus

Hozz létre egy saját típust `Time` néven. A típusnak legyen pontosan egy konstruktora `T` néven, amely két `Int` paramétert vár, amelyek az órát és a percet jelölik. Segítségül egyedül az egyenlőségvizsgálatot lehet a fordítótól kérni. (Tehát a `deriving` után csak az `Eq` állhat, semmi más!)

Ennek a felírása hasonlít a `Complex` típus felírásához, amit az óra végén írtam fel, tehát konstruktor neve és utána szóközzel elválasztva a paraméterek típusai kerülnek!

2. Okos konstruktor

Mivel Haskell-ben nem lehet olyat típusszinten megmondani, hogy mettől meddig lehetnek a `Time` típusban az értékek, tehát az óra `[0..23]`-ig, a perc pedig `[0..59]`-ig terjedhet, ezért hozzunk létre egy külön függvényt (ún. okos konstruktort), amely csak akkor fog visszaadni `Time` típusú értéket, ha mindkét érték a megadott intervallumba esik. Ha valamelyik érték nem megfelelő, a függvény az `error` függvénnyel dobjon hibát, mert jobb eszközünk még nincs.

```
t :: Int -> Int -> Time
```

3. Idő megjelenítése

Mivel a fordítótól nem lehetett kérni megjelenítést, ezért írjunk egyet magunknak, ami egy kicsit szebben fogja megjeleníteni az időt. Ehhez definiáljuk a `show` függvényt egy `instance Show Time where` után, ami paraméterül kap egy időt és `':'`-tal van elválasztva az óra és a perc.

```
instance Show Time where
    show ...
```

```
show (T 12 23) == "12:23"
show (T 1 1) == "1:1"
show (T 0 59) == "0:59"
```

Mivel nem akarok most a jó 0-ákkal szórakozni, azokat nem kell kiírni, tehát `"1:1"`-et kell kiírni a `"1:01"` helyett. (Persze lehet, aki olyat szeretne.)

4. Korábban, később

Definiálj egy függvényt `isEarlier` néven, amely két megadott időpont közül megmondja, hogy az első paraméterül kapott időpont előbb van-e, mint a második időpont.

```
isEarlier :: Time -> Time -> Bool
```

```
isEarlier (T 10 20) (T 10 21)
isEarlier (T 0 0) (T 23 59)
not $ isEarlier (T 21 55) (T 21 30)
not $ isEarlier (T 22 0) (T 21 30)
```

5. Két időpont között

Definiálj egy függvényt `isBetween` névvel, amely paraméterül kap 3 időpontot és vizsgálja, hogy a középső a két szélső időpont között van-e. Figyelni kell arra, hogy esetleg fordított sorrendben vannak az időpontok, arra is működnie kell. Tehát `t1 < t2 < t3` vagy `t3 < t2 < t1` teljesül-e.

```
isBetween :: Time -> Time -> Time -> Bool
```

```
isBetween (T 12 23) (T 12 34) (T 12 45)
isBetween (T 23 59) (T 12 0) (T 0 0)
not $ isBetween (T 20 0) (T 10 0) (T 16 0)
```

6. Amerikai idő típus

Definiálj egy új adattípust `USTime` néven! A típust úgy definiáld, hogy az Amerikában használatos AM, PM formátumnak helyet adjon, hiszen 12 órás óraformátumot használnak. Szintén kell bele az óra-perc paraméter. (Igen, szándékosan ilyen a megfogalmazás, kíváncsi vagyok, hogy ki hogy oldja meg.) Ebben is szintén kérhető az `Eq` és csak az!

Mivel előfordulhat, hogy mások a típusnak az értékei mindenkinél, a további tesztesetek csak iránymutatásra szolgálnak! Ebből kifolyólag az automatikus tesztelő csak eddig a pontig tudja ellenőrizni, hogy jó-e a megoldás, a továbbiakat én nézem meg kézzel. Tehát ha kiírja, hogy "Átment a teszteseteken", az messze nem jelenti azt, hogy jó; még kevésbé, mint a zh-n. Igyekezzetek kipróbálni a függvényeket, mielőtt megoldást küldtök be!

7. Amerikai idő létrehozása jól

Mivel 12 órás formátumot használnak, ezért hozz létre egy okos konstruktort `ustime` néven, amely ellenőrzi, hogy az órák az `[1..12]` intervallumban vannak (perc továbbra is `[0..59]`), és valami alapján el kell dönteni, hogy `AM` vagy `PM` az idő, ami mondjuk **ne** egy mezei `String` legyen, most már vannak jobb eszközeink.

```
ustime :: ... -> USTime
```

Megjegyzés: A megfelelő felépítést, típust nektek kell kitalálni.

8. Amerikai idő megjelenítés

Írjunk az amerikai időnek is egy `show` függvényt a fent látott módon, tehát:

```
instance Show USTime where
    show ...
```

Amely az alábbi módon jeleníti meg az időket (ugye írtam, hogy saját módon definiáld a USTime-ot, tehát attól függően kell oda a show-ban is írni a megfelelő adatokat):

```
show (délelőtti idő: 12 1) == "AM 12:1"
show (délutáni idő: 3 23) == "PM 3:23"
```

9a. Értelmezhető idő

Az amerikai óra a 0 órát és a delet is 12-vel jelöli. Tehát `0:11` náluk `AM 12:11`, `11:59` náluk `AM 11:59`, `12:00` náluk `PM 12:00`, tehát az órájuk abban a sorrendben megy, hogy 12,1,2,3,4,5,6,7,8,9,10,11. Ezek alapján alakítsd át a kapott amerikai időt rendes, normális, 24 órás időre.

```
ustimeToTime :: USTime -> Time
```

9b. Amerikainak is

Alakítsd át a rendes időt amerikaira.

```
timeToUSTime :: Time -> USTime
```

Figyelem! Az új értékek létrehozásához használjátok a megírt okos konstruktorokat, amik már ellenőrzik a helyességet! Az egész 9-es feladatra vonatkozik ez, nem csak a 'b' részére.

Tesztek

```
(t 12 23) == (T 12 23)
(t 20 45) == (T 20 45)
show (T 12 23) == "12:23"
show (T 1 1) == "1:1"
show (T 0 59) == "0:59"
isEarlier (T 10 20) (T 10 21)
isEarlier (T 0 0) (T 23 59)
isEarlier (T 10 10) (T 11 7)
not $ isEarlier (T 21 20) (T 21 20)
not $ isEarlier (T 21 55) (T 21 30)
not $ isEarlier (T 22 0) (T 21 30)
isBetween (T 12 23) (T 12 34) (T 12 45)
isBetween (T 12 10) (T 12 14) (T 14 9)
isBetween (T 11 14) (T 12 9) (T 12 40)
isBetween (T 11 10) (T 12 9) (T 13 8)
```

```
isBetween (T 23 59) (T 12 0) (T 0 0)
not $ isBetween (T 20 0) (T 10 0) (T 16 0)
```