A házi feladatot egy `Homework2` 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.)

Paramterikus polimorfizmus

- Definiáljuk a `triTuplify` függvényt amely három polimorf értéket egy tuple-be rak! (`triTuplify :: a -> b -> c -> (a,b,c)`)
- Definiáljuk a `replaceMiddle` függvényt amely egy tuple középső elemét lecseréli! (`replaceMiddle :: d -> (a,b,c) -> (a,d,c)`)
- Definiáljuk a `dropThird` függvényt amely egy tuple harmadik elemét eldobja! (`dropThird :: (a,b,c,d) -> (a,b,d)`)
- Definiáljuk az `identity` függvényt amely visszaadja a kapott paramétert eredményül! (`identity :: a -> a`)

Ad-hoc polimorfizmus

A feladat folyamán a megfelő kikötéseket kihagytam a típusszignatúrából, ezeket nektek kell kitalálni mi megy oda! Ismert típusosztályok: `Num`, `Eq`, `Ord`. Ha valamelyikről elfelejtetted mit tud, GHCi-ben a `:i` paranccsal meg tudod nézni (pl `:i Ord`). A megfelelő típusosztályt a `???` helyére rakjátok!

- Definiáljuk a `mulThree` függvényt amely három polimorf értéket összeszoroz! (`mulThree :: ??? a => a -> a -> a -> a`)
- Defiiáljuk a `comparison` függvényt amely két értékről eldönti a kapott paraméterekről, hogy nagyobb egyenlő-e illetve kisebb egyenlő-e és ezt egy tuple-ben visszaadja! (`comparison :: ??? a => a -> a -> (Bool {- első nagyobb egyenlő-e -} , Bool {- első kisebb egyenlő-e -})`)

Tesztek:

```
triTuplify 1 2 3 == (1,2,3)
triTuplify "alma" True 'c' == ("alma", True, 'c')
triTuplify (1,2) "barack" False == ((1,2), "barack", False)
replaceMiddle 1 ("alma", 'c', True) == ("alma", 1, True)
replaceMiddle "alma" (1, 2, 3) == (1, "alma", 3)
dropThird (1,2,3,4) == (1,2,4)
dropThird (1, 'c', 'a', True) == (1, 'c', True)
identity 1 == 1
(identity identity) 1 == 1
identity "barack" == "barack"
mulThree 1 2 3 == 6
mulThree 1.0 2.0 3.0 == 6.0
comparison 1 2 == (False, True)
comparison 1 1 == (True, True)
comparison () () == (True, True)
```

Nehezebb (nem kötelező) feladatok

Létezik haskellben egy `Fractional` típusosztály ami a nemegész számokra definiált műveleteket vonja össze. Egy polimorf típus `Fractional` megkötéssel megenged minden műveletet amit a `Num` is, és még az osztást is mellette a `/` operátorral. Emellet létezik egy `Floating` típusosztály ami azzal megkötött típusokra megengedi a trigonometriai függvények (`sin`, `cos`, `tan` stb) és a gyökvonás (`sqrt`) használatát.
A feladat során két elemű vektorokra fogunk műveleteket írni. A vektorok típusát `(a,a)` tuple-el fogjuk reprezentálni, ahol az `a` típusra `Floating` megkötést adunk meg. A típusszignatúrák itt nem lesznek megadva, ezeket ki kell találni.

- Definiáljunk alapműveleteket vektorokra! (`vecAdd` = vektorok közti összeadás, `vecSub` = vektorok közti kivonás, `vecCMul` = vektor számmal való szorzása, `vecCDiv` = vektor számmal való osztása)
- Definiáljunk egy `arg` nevű függvényt ami egy paraméterül kapott vektor X tengellyel bezárt szögét kiszámolja. Az erra használható formula: `arctan(y/x) / pi * 180`