Házi feladat

A házi feladatot egy Homework1 nevű modulként kell beadni. Minden definiálandó függvényhez adjuk meg a hozzá tartozó típusszignatúrát is! A feladatok után a zárójelben lévő név azt jelzi, milyen néven kell definiálni az adott függvényt, kifejezést. A forrásfáljban ügyeljetek arra, hogy minden kifejezés rendelkezzen helyes típusszignatúrával!

3D Vektor

Definiáljuk a `Vector3` adattípust, mely 3 dimenziós vektorok ábrázolására lesz használatos. Az egyetlen adatkonstruktor legyen `V`, és legyen három `Int` adattagja, mint a három komponens. Kérjük a fordítótól a `Show` és az `Eq` típusosztály automatikus példányosítását.

Komponensek összege

Adjuk össze három komponensét egy vektornak!

```
componentSum :: Vector3 -> Int
```

Vektoriális szorzat

Számoljuk ki két, paraméterként kapott vektor vektoriális szorzatát! `crossProduct :: Vector3 -> Vector3 -> Vector3`

Két vektor vektoriális szorzata az alábbi formulából számolható:

$c_1 = a_2b_3 - a_3b_2$

$c_2 = a_3b_1 - a_1b_3$

$c_3 = a_1b_2 - a_2b_1$

Vektor-lista összegzése

Adjuk össze a vektorokat egy listában! Két vektort úgy adunk össze, hogy az azonos komponenseit összeadjuk.

```
vectorListSum :: [Vector3] -> Vector3
```

Háttértár

Definiáld a `Storage` adattípust, ami segítségével különböző háttértárakat tudunk jellemezni. Két konstruktora legyen, az egyik legyen a `HDD`. Az első adattagja a gyártójának neve legyen, a második a percenkénti fordulatszáma, `Int` típusú értékként tárolva, az utolsó a kapacitása legyen GB-ban megadva, ugyanúgy `Int` típusú értékként tárolva. A másik konstruktora az `SSD` legyen, ahol tároljuk a gyártóját és a kapacitását. Kérjük a fordítótól a `Show` és az `Eq` típusosztály automatikus példányosítását.

Kapacitás

Írjunk egy függvényt, ami megadja egy háttértár kapacítását.

```
capacity :: Storage -> Int
```

HDD vagy SSD

Adjuk meg azt a függvényt, ami eldönti, hogy egy háttértár `HDD`-e.

```
isHDD :: Storage -> Bool
```

Legnagyobbnál is nagyobbak

Adjuk meg azt a függvényt, ami egy háttértár listából visszaadja az összes olyan `HDD`, aminek nagyobb a kapacítása, mint a legnagyobb kapacítású `SSD`-nek. A feladat megoldásához érdemes segényfüggvényt használni.

```
hugeHDDs :: [Storage] -> [Storage]
```

Az alábbi tesztesetek közül mindegyiknek `True`-t kell adnia:

```
componentSum (V 4 6 7) == 17
componentSum  (V 0 0 0) == 0
componentSum  (V 1 1 1) == 3
componentSum  (V 1 (-3) 10) == 8
componentSum  (V 6 1 10) == 17
componentSum (V (-1) 45 (-55)) == (-11)
crossProduct (V 5 6 7) (V 3 4 5) == V 2 (-4) 2
crossProduct (V 1 2 3) (V 3 2 1) == V (-4) 8 (-4)
crossProduct (V 1 1 8) (V 0 2 1) == V (-15) (-1) 2
crossProduct (V 7 4 3) (V 8 3 1) == V (-5) 17 (-11)
vectorListSum [] == V 0 0 0
vectorListSum [V 1 2 3] == V 1 2 3
vectorListSum [V 1 1 1, V 0 0 0, V 2 2 3] == V 3 3 4
vectorListSum [V 0 2 4, V (-10) 4 2, V 0 0 0] == V (-10) 6 6
vectorListSum [V 0 0 0, V 0 0 0, V 0 0 0, V 0 0 0] == V 0 0 0
vectorListSum [V i (i*i) (i*i*i) | i <- [0..10]] == V 55 385 3025
capacity (HDD "Seagate" 5600 250) == 250
capacity (HDD "Verbatim" 7200 1000) == 1000
capacity (SSD "Samsung" 500) == 500
capacity (SSD "Samsung" 750) == 750
isHDD (HDD "Seagate" 5600 250)
isHDD (HDD "Verbatim" 7200 1000)
not $ isHDD (SSD "Samsung" 500)
not $ isHDD (SSD "Samsung" 750)
hugeHDDs [] == []
hugeHDDs [HDD "Seagate" 5600 250, HDD "Verbatim" 7200 1000, SSD "Samsung" 500,
```

```
SSD "Samsung" 750] == [HDD "Verbatim" 7200 1000]
hugeHDDs [HDD "Seagate" 5600 250, HDD "Verbatim" 7200 100,
SSD "Samsung" 250, SSD "Samsung" 100] == []
hugeHDDs [HDD "Seagate" 5600 1000, HDD "Verbatim" 7200 1500, SSD "Samsung" 250,
SSD "Samsung" 500] == [HDD "Seagate" 5600 1000,HDD "Verbatim" 7200 1500]
```