Rekurzió, elágazás

Most már vannak elágazások, így van néhány felhívás a használhatukkal kapcsolatban:

- Ha egy függvény eredménye `Bool` típusú kell legyen, akkor az elágazás teljesen felesleges dolog, így nem is lehet használni azokban a függvényekben.
- Egy `Bool` típusú értéket egyenlőségvizsgálni megint teljesen felesleges (pl. `(1 > 2) == True`, már az `1 > 2` megválaszolja ugyanazt), így szintén nem szabad úgy használni, csúnya dolog, redundáns, attól csak feleslegesen több a kód, nem jobb.
- `if-then-else` továbbra sem használható! Tessék megszokni, hogy vannak őrfeltételek.

Más is van, amit eddig talán nem tisztáztam (listás dolgok):

- Haskellben a végtelen lista az alapértelmezett feltevés, ettől csak akkor lehet eltérni, ha a feladat kifejezetten kiemeli, hogy csak véges listára lehet/kell/fog működni az adott függvény.
- Listagenerátorban ha nem csinálunk semmit az elemmel és nincs is feltétel, akkor maga a generátor egy teljesen felesleges és nem szép dolog használni (`[c | c <- xs]` helyett ez csak simán `xs`).
- Tessék az elnevező (`@`) operátort használni, ahol van értelme.
- Ha már mintaillesztettünk a lista első és maradék elemére, tessék azokat használni és nem azt írni, hogy `head (x:xs)`, mert szintén semmi értelme nincs, csak feleslegesen sok kód, ez `x` maga közvetlen.

Talán sikerült mindent leírnom; ezek az általam kényszerített szabályok abban segítenek, hogy szebb kódot írjatok, akár más nyelven is (pl. `c++`-ban is teljesen felesleges `Bool` értéket egyenlőségvizsgálni).

0. Modul

Hozz létre egy modult `Hazi5` néven!

1. Rendezett-e

Definiáld az `isSorted` nevű függvényt, amely ellenőrzi rendezhető elemek listájáról, hogy az elemei növekvő sorrendben vannak-e. Ha a lista növekvően rendezett, akkor végtelen lista esetén a függvény nem terminál (remélhetőleg értelemszerű okok miatt).

```
isSorted ([] :: [Integer])
isSorted [1::Int]
isSorted [1::Integer]
isSorted [1::Double]
isSorted "a"
isSorted [5,6,9,10]
isSorted [(-2),(-1),1,9,10,19]
isSorted "adn"
not (isSorted [10,9,8,7,6,5,4,3,2,1,0])
not (isSorted "alma")
not (isSorted [10,9..])
not (isSorted ([1..10] ++ [9,8..]))
```

2. Indexelés javítása

Készítsd el a `(!!!)` függvényt, amely segítségével bele tudunk indexelni egy listába 0-tól kezdve az indexelést. Ha negatív indexet kap a függvény, akkor hátulról adjuk vissza az elemet (feltehető, hogy ebben és csak is ebben az

esetben a lista véges). A legutolsó elem a `(-1)`-es indextől kezdődik. **Ne** használd a `(!!)` operátort a definícióban! Használj mintaillesztést és rekurziót, `take`, `drop`, `length` és általánosabb variánsaik teljesen feleslegesek, épp ezért nem használhatóak!

Segítség: A hátulról haladáshoz érdemes a `reverse` függvényt használni egyszer, mert akkor már ugyanúgy elölről lehet haldni. `reverse :: [a] -> [a]`, megfordítja egy véges lista elemeinek sorrendjét.

```
[0..] !!! (136 :: Integer) == 136
"alma" !!! (2 :: Int) == 'm'
[False, True, False, False, True, False] !!! ((-5) :: Integer)
[0..10] !!! (-1) == 10
[0..10] !!! (-11) == 0
[0..10] !!! 0 == 0
-- [0..10] !!! (-12) -> error "Túl kicsi index!"
-- [0..10] !!! 11 -> error "Túl nagy index!"
-- [0..] !!! (-1) -> végtelenségig fut.
```

(A nyilak nem a konkrét visszatérési értéket mutatják, hanem azt, hogy minek kell történnie azokra az értékekre, amolyan iránymutatás. Az (==) viszont konkrét érték, annak mindig mindenkor teljesülnie kell.)

3. Format javítása

Az előző házi `format` függvényét egészítsd ki úgy, hogy a negatív számokra is működjön! A függvény neve maradjon `format` ugyanúgy. Értelemszerűen negatív hosszra nehéz kiegészíteni; ha negatív a szám, akkor azt csinálja, mintha `0`-t kapott volna paraméterül.

4. Viharos vidék

Adott egy régió, összes meteorológiai állomásának mérései. Rendezett 4-esekben tároljuk sorban a mérőállomás nevét, a hőmérsékletet °C-ban megadva, a szél sebességét km/h-ban megadva és hogy az adott napon hányadik rögzített adatot küldi. Definiálj egy függvényt `mightyGale` néven, amely megadja az első mérőállomás nevét, ahol több mint 110 km/h, azaz orkán erejű szél volt. Ha nem volt sehol sem orkán erejű szél, az eredmény legyen üres `String`!

Segítség: A típusba sok megkötés fog kelleni az egyes adatokra vonatkozóan, hiszen azok egymástól függetlenek.

```
mightyGale [("Pest6", 23.2 :: Double, 54.5 :: Float, 2 :: Int), ("Buda2", 21.1, 77.7, 5), ("KPest1", 19.8
mightyGale [("Kamut1", 19 :: Integer, 55 :: Int, 1 :: Integer), ("Szentes2", 18, 112, 2), ("Cegled3", 18,
mightyGale [("Szergeny3", 23 :: Integer, 1 :: Double, 1 :: Int), ("Nagytevel1", 30, 12, 5), ("Himod4", 1,
mightyGale [("Jaszbereny1", 19.1 :: Float, 55 :: Integer, 1), ("Jaszbereny2", 18.9, 57, 2), ("Jaszfelsosz
mightyGale [("Sopron1", 0, 100, 1), ("Szekesfehervar2", 1, 110, 10), ("Siofok1", (-1), 110.1, 3)] == "Sio
```

5. Titkok tudója

Definiáld a `cipher` függvényt, amely egy titkosított szövegből kinyeri az első olyan kettő hosszú karaktersort, amelyet számjegy követ. Ha nincs ilyen, akkor az eredmény legyen üres `String`. A megoldásban használj mintaillesztést és rekurziót! `take`, `drop`, `length` és általánosabb variánsaik teljesen feleslegesek, épp ezért nem használhatóak!

Segítség: Az egyes karakterek azonosításához használjuk a Data.Char függvényeit. Hogy melyiket? Azt kell nektek megtalálni.

```
cipher "PYdg7iT4vdO0n4AgmGfUpRzogAf" == "dg"
cipher "PYdgaiTLvdOKnAAgmGfUpRzogA4" == "gA"
cipher "4vkYyAO174midQTt0" == "AO"
cipher "BwxwEwqCKHuMTAaPn" == ""
cipher ['\0'..] == "./"
cipher "dM7" == "dM"
cipher "777" == "77"
cipher "Kmz" == ""
cipher "Zk"  == ""
cipher "T4"  == ""
cipher "" == ""
```

6. Dupla elemek

Definiáld a `doubleElements` függvényt, amely minden egyes lista elemét egyenként megkettőzi egymás után! Használj rekurziót!

```
doubleElements [1,2,3] == [1,1,2,2,3,3]
null (doubleElements [])
doubleElements "alma" == "aallmmaa"
take 10 (doubleElements [0..]) == [0,0,1,1,2,2,3,3,4,4]
```

7. Sok szóköz

Definiáld a `deleteDuplicateSpaces` függvényt, amely egy szövegből eltávolítja a több egymás mellett álló szóközöket, azokból egyet meghagyva. A szöveg eleji és szöveg legvégi szóközöket teljes egészében dobja el a függvény.

```
deleteDuplicateSpaces "alma  szilva    barack" == "alma szilva barack"
deleteDuplicateSpaces "    alma  szilva    barack    eper   " == "alma szilva barack eper"
deleteDuplicateSpaces "    alma  szilva    barack    eper" == "alma szilva barack eper"
deleteDuplicateSpaces "    alma  szilva    barack    eper " == "alma szilva barack eper"
deleteDuplicateSpaces "alma  szilva    barack    eper  " == "alma szilva barack eper"
deleteDuplicateSpaces " alma  szilva    barack    eper   " == "alma szilva barack eper"
take 12 (deleteDuplicateSpaces (cycle "a   b")) == "a ba ba ba b"
null (deleteDuplicateSpaces "")
```

Tesztelés

```
allPassed :: Bool
allPassed = null allTests

allTests :: [(String, Bool)]
allTests = [(str,test) |
   (str, test) <- [
```

```
    ("isSorted ([] :: [Integer])", isSorted ([] :: [Integer]))
  , ("isSorted [1::Int]", isSorted [1::Int])
  , ("isSorted [1::Integer]", isSorted [1::Integer])
  , ("isSorted [1::Double]", isSorted [1::Double])
  , ("isSorted \"a\"", isSorted "a")
  , ("isSorted [5,6,9,10]", isSorted [5,6,9,10])
  , ("isSorted [(-2),(-1),1,9,10,19]", isSorted [(-2),(-1),1,9,10,19])
  , ("isSorted \"adn\"", isSorted "adn")
  , ("not (isSorted [10,9,8,7,6,5,4,3,2,1,0])", not (isSorted [10,9,8,7,6,5,4,3,2,1,0]))
  , ("not (isSorted \"alma\")", not (isSorted "alma"))
  , ("not (isSorted [10,9..])", not (isSorted [10,9..]))
  , ("not (isSorted ([1..10] ++ [9,8..]))", not (isSorted ([1..10] ++ [9,8..])))
  , ("[0..] !!! (136 :: Integer) == 136", [0..] !!! (136 :: Integer) == 136)
  , ("\"alma\" !!! (2 :: Int) == 'm'", "alma" !!! (2 :: Int) == 'm')
  , ("[False, True, False, False, True, False] !!! ((-5) :: Integer)", [False, True, False, False, Tr
  , ("[0..10] !!! (-1) == 10", [0..10] !!! (-1) == 10)
  , ("[0..10] !!! (-11) == 0", [0..10] !!! (-11) == 0)
  , ("[0..10] !!! 0 == 0", [0..10] !!! 0 == 0)
  , ("format (10 :: Int) \"alma\" == \"alma      \"", format (10 :: Int) "alma" == "alma      ")
  , ("format (4 :: Integer) \"\" == \"    \"", format (4 :: Integer) "" == "    ")
  , ("format (5 :: Integer) \"szilva\" == \"szilva\"", format (5 :: Integer) "szilva" == "szilva")
  , ("format (0 :: Int) \"\" == \"\"", format (0 :: Int) "" == "")
  , ("format (0 :: Integer) \"barack\" == \"barack\"", format (0 :: Integer) "barack" == "barack")
  , ("take 50 (format (60 :: Integer) (repeat 'a')) == replicate 50 'a'", take 50 (format (60 :: Inte
  , ("format ((-10) :: Int) \"alma\" == \"alma\"", format ((-10) :: Int) "alma" == "alma")
  , ("format ((-4) :: Integer) \"\" == \"\"", format ((-4) :: Integer) "" == "")
  , ("format ((-5) :: Integer) \"szilva\" == \"szilva\"", format ((-5) :: Integer) "szilva" == "szilv
  , ("take 50 (format ((-60) :: Integer) (repeat 'a')) == replicate 50 'a'", take 50 (format ((-60) :
  , ("mightyGale [(\"Pest6\", 23.2 :: Double, 54.5 :: Float, 2 :: Int), (\"Buda2\", 21.1, 77.7, 5), (
  , ("mightyGale [(\"Kamut1\", 19 :: Integer, 55 :: Int, 1 :: Integer), (\"Szentes2\", 18, 112, 2), (
  , ("mightyGale [(\"Szergeny3\", 23 :: Integer, 1 :: Double, 1 :: Int), (\"Nagytevel1\", 30, 12, 5),
  , ("mightyGale [(\"Jaszbereny1\", 19.1 :: Float, 55 :: Integer, 1), (\"Jaszbereny2\", 18.9, 57, 2),
  , ("mightyGale [(\"Sopron1\", 0, 100, 1), (\"Szekesfehervar2\", 1, 110, 10), (\"Siofok1\", (-1), 11
  , ("cipher \"PYdg7iT4vdO0n4AgmGfUpRzogAf\" == \"dg\"", cipher "PYdg7iT4vdO0n4AgmGfUpRzogAf" == "dg"
  , ("cipher \"PYdgaiTLvdOKnAAgmGfUpRzogA4\" == \"gA\"", cipher "PYdgaiTLvdOKnAAgmGfUpRzogA4" == "gA"
  , ("cipher \"4vkYyAO174midQTt0\" == \"AO\"", cipher "4vkYyAO174midQTt0" == "AO")
  , ("cipher \"BwxwEwqCKHuMTAaPn\" == \"\"", cipher "BwxwEwqCKHuMTAaPn" == "")
  , ("cipher ['\0'..] == \"./\"", cipher ['\0'..] == "./")
  , ("cipher \"dM7\" == \"dM\"", cipher "dM7" == "dM")
  , ("cipher \"777\" == \"77\"", cipher "777" == "77")
  , ("cipher \"Kmz\" == \"\"", cipher "Kmz" == "")
  , ("cipher \"Zk\"  == \"\"", cipher "Zk"  == "")
  , ("cipher \"T4\"  == \"\"", cipher "T4"  == "")
  , ("cipher \"\" == \"\"", cipher "" == "")
  , ("doubleElements [1,2,3] == [1,1,2,2,3,3]", doubleElements [1,2,3] == [1,1,2,2,3,3])
  , ("null (doubleElements [])", null (doubleElements []))
  , ("doubleElements \"alma\" == \"aallmmaa\"", doubleElements "alma" == "aallmmaa")
  , ("take 10 (doubleElements [0..]) == [0,0,1,1,2,2,3,3,4,4]", take 10 (doubleElements [0..]) == [0,
  , ("deleteDuplicateSpaces \"alma  szilva    barack\" == \"alma szilva barack\"", deleteDuplicateSpa
  , ("deleteDuplicateSpaces \"    alma  szilva    barack    eper   \" == \"alma szilva barack eper\""
  , ("deleteDuplicateSpaces \"    alma  szilva    barack    eper\" == \"alma szilva barack eper\"", d
  , ("deleteDuplicateSpaces \"    alma  szilva    barack    eper \" == \"alma szilva barack eper\",
  , ("deleteDuplicateSpaces \"alma  szilva    barack    eper  \" == \"alma szilva barack eper\"", del
  , ("deleteDuplicateSpaces \" alma  szilva    barack    eper   \" == \"alma szilva barack eper\"", d
  , ("take 12 (deleteDuplicateSpaces (cycle \"a   b\")) == \"a ba ba ba b\"", take 12 (deleteDuplicat
  , ("null (deleteDuplicateSpaces \"\")", null (deleteDuplicateSpaces ""))
], not test]
```

Összes teszt

```
isSorted ([] :: [Integer])
isSorted [1::Int]
isSorted [1::Integer]
isSorted [1::Double]
isSorted "a"
isSorted [5,6,9,10]
isSorted [(-2),(-1),1,9,10,19]
isSorted "adn"
not (isSorted [10,9,8,7,6,5,4,3,2,1,0])
not (isSorted "alma")
not (isSorted [10,9..])
not (isSorted ([1..10] ++ [9,8..]))
[0..] !!! (136 :: Integer) == 136
"alma" !!! (2 :: Int) == 'm'
[False, True, False, False, True, False] !!! ((-5) :: Integer)
[0..10] !!! (-1) == 10
[0..10] !!! (-11) == 0
[0..10] !!! 0 == 0
format (10 :: Int) "alma" == "alma      "
format (4 :: Integer) "" == "    "
format (5 :: Integer) "szilva" == "szilva"
format (0 :: Int) "" == ""
format (0 :: Integer) "barack" == "barack"
take 50 (format (60 :: Integer) (repeat 'a')) == replicate 50 'a'
format ((-10) :: Int) "alma" == "alma"
format ((-4) :: Integer) "" == ""
format ((-5) :: Integer) "szilva" == "szilva"
take 50 (format ((-60) :: Integer) (repeat 'a')) == replicate 50 'a'
mightyGale [("Pest6", 23.2 :: Double, 54.5 :: Float, 2 :: Int), ("Buda2", 21.1, 77.7, 5), ("KPest1", 19.8
mightyGale [("Kamut1", 19 :: Integer, 55 :: Int, 1 :: Integer), ("Szentes2", 18, 112, 2), ("Cegled3", 18,
mightyGale [("Szergeny3", 23 :: Integer, 1 :: Double, 1 :: Int), ("Nagytevel1", 30, 12, 5), ("Himod4", 1,
mightyGale [("Jaszbereny1", 19.1 :: Float, 55 :: Integer, 1), ("Jaszbereny2", 18.9, 57, 2), ("Jaszfelsosz
mightyGale [("Sopron1", 0, 100, 1), ("Szekesfehervar2", 1, 110, 10), ("Siofok1", (-1), 110.1, 3)] == "Sio
cipher "PYdg7iT4vdO0n4AgmGfUpRzogAf" == "dg"
cipher "PYdgaiTLvdOKnAAgmGfUpRzogA4" == "gA"
cipher "4vkYyAO174midQTt0" == "AO"
cipher "BwxwEwqCKHuMTAaPn" == ""
cipher ['\0'..] == "./"
cipher "dM7" == "dM"
cipher "777" == "77"
cipher "Kmz" == ""
cipher "Zk"  == ""
cipher "T4"  == ""
cipher "" == ""
doubleElements [1,2,3] == [1,1,2,2,3,3]
null (doubleElements [])
doubleElements "alma" == "aallmmaa"
take 10 (doubleElements [0..]) == [0,0,1,1,2,2,3,3,4,4]
deleteDuplicateSpaces "alma  szilva    barack" == "alma szilva barack"
deleteDuplicateSpaces "    alma  szilva    barack    eper   " == "alma szilva barack eper"
deleteDuplicateSpaces "    alma  szilva    barack    eper" == "alma szilva barack eper"
deleteDuplicateSpaces "    alma  szilva    barack    eper " == "alma szilva barack eper"
deleteDuplicateSpaces "alma  szilva    barack    eper  " == "alma szilva barack eper"
deleteDuplicateSpaces " alma  szilva    barack    eper   " == "alma szilva barack eper"
```

```
take 12 (deleteDuplicateSpaces (cycle "a   b")) == "a ba ba ba b"
null (deleteDuplicateSpaces "")
```