Csoportok

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 12

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 91

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 92

2. ZH - A csoport

Kategória: Vizsgafeladatok
Elérhető: 2022. 10. 21. 8:15
Pótolható határidő:
Végső határidő: 2022. 10. 21. 8:43
Kiírta: Erdei Zsófia

Leírás:

Funkcionális programozás 2. ZH - A csoport

Előzetes tudnivalók

Használható segédanyagok:

- Haskell könyvtárak dokumentációja,
- Hoogle,
- a tárgy honlapja, és a
- Haskell szintaxis összefoglaló.

Más segítőeszköz nem használható.

Ha bármilyen kérdés, észrevétel felmerül, azt a gyakorlatvezetőnek kell jelezni, **nem** a diáktársaknak!

A feladatsor megoldására 25 perc áll rendelkezésre

A feladatok tetszőleges sorrendben megoldhatóak. A pontozás szabályai a következők:

- Minden teszten átmenő megoldás ér teljes pontszámot.
- Funkcionálisan hibás (valamelyik teszteseten megbukó) megoldás nem ér pontot.
- Fordítási hibás vagy hiányzó megoldás esetén a teljes megoldás 0 pontos.

Ha hiányos/hibás részek lennének a feltöltött megoldásban, azok kommentben szerepeljenek.

Tekintve, hogy a tesztesetek, bár odafigyelés mellett íródnak, nem fedik le minden esetben a függvény teljes működését, határozottan javasolt még külön próbálgatni a megoldásokat beadás előtt!

Az elméleti kérdésekre adott választ a forráskódban kell elhelyezni, kommentben. Minden függvénynek meg kell adni a típusszignatúráját is. A függvények elvárt neve és típusa zárójelben meg van adva.

`Zarthelyi2` néven kell deklarálni a modult. A `.hs` fájlt `.zip` -be tömörítve kell beadni.

Legalább egy feladatot rekurzívan kell megoldani!

Elméleti kérdések

1. Adott az alábbi kód. Miért nem lehet kiértékelni az `f []` -t? (1 pont)

```
f :: Num a => [a] -> a
f (x:xs) = 5
```

2. Adj egy példa kifejezést, amely illeszkedik az alábbi mintára: `[(x,[]),_]` ! (1 pont)

Feladatok

Minden második elem duplázása (1 pont)

Definiáld a `duplicateEverySecond :: [a] -> [a]` függvényt, amely a paraméterként kapott listából egy olyan listát állít elő, amely az eredeti lista minden második elemét kétszer tartalmazza!

```
duplicateEverySecond [] == []
duplicateEverySecond [1] == [1]
duplicateEverySecond [1] == [1]
duplicateEverySecond [1,2] == [1,2,2]
duplicateEverySecond [1..9] == [1,2,2,3,4,4,5,6,6,7,8,8,9]
duplicateEverySecond [1..10] == [1,2,2,3,4,4,5,6,6,7,8,8,9,10,10]
```

Növekvő párok (1 pont)

Definiáld a `sortTuples :: Ord a => [(a,a)] -> [(a,a)]` függvényt, amely egy rendezett párokat tartalmazó listából egy olyan listát állít elő, ahol a rendezett párok eredeti komponensei növekvő sorrendben szerepelnek!

```
sortTuples [] == []
sortTuples [(2,1)] == [(1,2)]
sortTuples [(2,1),(1,1),(1,2)] == [(1,2),(1,1),(1,2)]
```

Összefon (2 pont)

Definiáld az `interleave :: [a] -> [a] -> [a]` függvényt, amely két listát "összefon" olyan módon, hogy az eredmény lista a bemenetek elemeit tartalmazza váltakozva. Feltehetjük, hogy a paraméterként kapott listák azonos hosszúak.

```
interleave [] [] == []
interleave [1] [2] == [1,2]
interleave [0,0,0,0] [1,1,1,1] == [0,1,0,1,0,1,0,1]
interleave [1..5] [6..10] == [1,6,2,7,3,8,4,9,5,10]
```