Csoportok

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 12

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 91

Eseményvezérelt alkalmazások
IP-18bEVALKEG | 92

2. ZH - B 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 - B 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 20 perc áll rendelkezésre (+ 2 perc feltöltésre)

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. Adj egy példa kifejezést, amely illeszkedik az alábbi mintára: `((x:y),[z])` ! (1 pont)
2. Mi a különbség a `null x` és a `length x == 0` között működésükben? (1 pont)

Feladatok

Növekvő párok (1 pont)

Definiáld a `keepIncreasingTriples :: Ord a => [(a,a,a)] -> [(a,a,a)]` függvényt, amely egy rendezett hármasokat tartalmazó listából csak azokat tartja meg, ahol a komponensek szigorúan monoton növekvő sorrendben szerepelnek!

```
keepIncreasingTriples [] == []
keepIncreasingTriples [(1,2,3)] == [(1,2,3)]
keepIncreasingTriples [(1,2,2)] == []
keepIncreasingTriples [(1,2,3),(1,2,2),(1,5,9),(9,7,4)] == [(1,2,3),(1,5,9)]
```

Minden harmadik elem törlése (1 pont)

Definiáld a `deleteEveryThird :: [a] -> [a]` függvényt, amely a paraméterként kapott listából töröl minden harmadik elemet!

```
deleteEveryThird [] == []
deleteEveryThird [1] == [1]
deleteEveryThird [1,2] == [1,2]
deleteEveryThird [1,2,3] == [1,2]
deleteEveryThird [1..10] == [1,2,4,5,7,8,10]
```

Váltakozó elemek (2 pont)

Definiáld az `alternate :: [a] -> [a] -> [a]` függvényt, amely a paraméterül kapott két listából előállít egy olyan listát, amely váltakozva tartalmazza a bemeneti listák elemeit olyan módon, hogy első lista minden páratlan indexű elemét és a második lista minden páros indexű elemét tartalmazza. Feltehetjük, hogy a paraméterként kapott listák azonos hosszúak.

```
alternate [] [] == []
alternate [1] [2] == [1]
alternate [1,2] [3,4] == [1,4]
alternate [1,2,3] [4,5,6] == [1,5,3]
alternate [1..5] [0,0,0,0,0] == [1,0,3,0,5]
alternate [1..6] [0,0,0,0,0,0] == [1,0,3,0,5,0]
```

Megoldás

Letöltés

Név: Zarthelyi2.zip
Feltöltés ideje: 2022. 10. 21. 8:37
Értékelés:
Státusz: Elfogadva
Feltöltések száma: 1
Értékelte: Erdei Zsófia
Megjegyzések: 0.0 1.0

Automatikus tesztelés eredményei

```
Valamelyik tesztesetre hibásan futott le a beadott program.

Megbukott tesztek:

## 2.
## Kivétel:
submission:(8,1)-(11,20): Non-exhaustive patterns in function keepIncreasingTriples

## Teszteset:
keepIncreasingTriples [(1, 2, 3)] == [(1, 2, 3)]


## 3.
## Kivétel:
submission:(8,1)-(11,20): Non-exhaustive patterns in function keepIncreasingTriples

## Teszteset:
keepIncreasingTriples [(1, 2, 2)] == []


## 4.
## Eredmény:
False

## Teszteset:
keepIncreasingTriples [(1, 2, 3), (1, 2, 2), (1, 5, 9), (9, 7, 4)]
  == [(1, 2, 3), (1, 5, 9)]


## 7.
## Kivétel:
submission:(14,1)-(16,57): Non-exhaustive patterns in function deleteEveryThird

## Teszteset:
deleteEveryThird [1, 2] == [1, 2]


## 10.
## Kivétel:
Eval.hs:61:401: error:
    Variable not in scope: alternate :: [a0] -> [a1] -> [()]
(deferred type error)

## Teszteset:
alternate [] [] == []


## 11.
## Kivétel:
Eval.hs:61:423: error:
    Variable not in scope:
      alternate :: [Integer] -> [Integer] -> [Integer]
(deferred type error)

## Teszteset:
alternate [1] [2] == [1]


## 12.
## Kivétel:
Eval.hs:61:448: error:
    Variable not in scope:
      alternate :: [Integer] -> [Integer] -> [Integer]
(deferred type error)

## Teszteset:
alternate [1, 2] [3, 4] == [1, 4]


## 13.
## Kivétel:
Eval.hs:61:479: error:
    Variable not in scope:
      alternate :: [Integer] -> [Integer] -> [Integer]
(deferred type error)

## Teszteset:
alternate [1, 2, 3] [4, 5, 6] == [1, 5, 3]
```

```
## 14.
## Kivétel:
Eval.hs:61:516: error:
    Variable not in scope:
      alternate :: [Integer] -> [Integer] -> [Integer]
(deferred type error)

## Teszteset:
alternate [1 .. 5] [0, 0, 0, 0, 0] == [1, 0, 3, 0, 5]


## 15.
## Kivétel:
Eval.hs:61:560: error:
    Variable not in scope:
      alternate :: [Integer] -> [Integer] -> [Integer]
(deferred type error)

## Teszteset:
alternate [1 .. 6] [0, 0, 0, 0, 0, 0] == [1, 0, 3, 0, 5, 0]
```