Előzetes tudnivalók

Használható segédanyagok:

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

Más segédeszköz nem használható.

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

A feladatsor megoldására 15 perc áll rendelkezésre (+ 5 perc feltöltésre)

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

- Minden teszten átmenő, a feladat kikötéseinek megfelelő megoldás ér teljes pontszámot.
- Funkcionálisan hibás (valamelyik teszteseten megbukó) megoldás nem ér pontot.
- Fordítási hibás kód esetén a teljes zh 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 válaszokat a forráskódban kell elhelyezni, kommentben. Minden függvénynek meg kell adni a típusszignatúráját is. A függvények elvárt neve és típusa meg van adva. `Zarthelyi1` néven kell deklarálni a modult. A `.hs` fájlt `.zip`-be tömörítve kell beadni.

Elméleti kérdések (1 pont / kérdés)

- Mik a "totális függvények"?
- Adott az alábbi függvény:

```
f :: (Int, Double) -> Double
f (x, y) = x * (y * (-1))
```

Helyes-e az `f` függvény? Ha igen, mi lesz az eredmény `f (1,2)` esetén? Ha nem, mi a hiba vele és hogyan lehetne kijavítani úgy, hogy a "lényegi" működés ne változzon?

Gyakorlati feladatok

Három szám négyzetösszege (1 pont)

Definiáljunk azt a függvényt, amely egy rendezett hármasból, egy rendezett négyest ad vissza, ahol a számnégyes első három komponense a kapott hármas értékeivel egyezik meg, a negyedik komponens pedig a három szám négyzetösszege legyen!

```
squareSum :: Num a => (a,a,a) -> (a,a,a,a)
```

```
squareSum (1,1,1) == (1,1,1,3)
squareSum (-3,3,2) == (-3,3,2,22)
squareSum (-1,-1,-1) == (-1,-1,-1,3)
squareSum (-3,-3,0) == (-3,-3,0,18)
```

Vagy! És? (1 pont)

Adjuk meg azt a függvényt, amely az alábbi logikai állítást értékeli ki: ((A vagy B) és C). Használjunk mintaillesztést! A megoldás legyen a harmadik paraméterében lusta. (Tehát a harmadik paraméteren előfordulhatnak Exception-t okozó kifejezések.) Az alábbi igazságtábla segítségünkre jöhet.

A	B	C	ORAND(A,B,C)
Igaz	Igaz	Igaz	Igaz
Igaz	Igaz	Hamis	Hamis
Igaz	Hamis	Igaz	Igaz
Igaz	Hamis	Hamis	Hamis
Hamis	Igaz	Igaz	Igaz
Hamis	Igaz	Hamis	Hamis
Hamis	Hamis	Igaz	Hamis
Hamis	Hamis	Hamis	Hamis

```
orAnd :: Bool -> Bool -> Bool -> Bool
```

```
orAnd True True True
not (orAnd True True False)
orAnd True False True
not (orAnd (1 == 0) False (10 `div` 0 == 10))
```

Kétszázhúsz felett (2 pont)

Adott egy egész számokból álló lista. Határozzuk meg, hogy tartalmaz-e 220-nál nagyobb páros számot!

```
hasHugeEven :: [Integer] -> Bool
```

```
not (hasHugeEven [])
not (hasHugeEven [221,0,2])
not (hasHugeEven [220,0,2])
hasHugeEven [222,0,2]
hasHugeEven [222]
hasHugeEven [1,2,221,222]
```