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 35 perc áll rendelkezésre (+ 5 perc feltöltésre)

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

- Minden teszten átmenő, a feladat kikötéseinek megfelelő megoldás érhet 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. `Zarthelyi4Pot` néven kell deklarálni a modult. A `.hs` fájlt `.zip`-be tömörítve kell beadni.

Elméleti kérdés (1 pont)

Mit jelent a rekurzió?

Gyakorlati feladatok (5 pont)

Kártyák (1 pont)

Definiáljunk típusokat a kártyák reprezentálására.

A franciakártyában minden kártyát két tulajdonsággal azonosítunk: a színűkkel és a értékükkel.

- Vezess be `Rank` néven egy típusszinonimát az érték reprezentálására. Az érték `Int` típusú egész számokkal reprezentálható.

- Definiálj `Suit` néven egy adattípust a színek reprezentálására. A franciakártyában négy színt különböztethetünk meg, ennek megfelelően a színek típusának négy adatkonstruktora lesz:

  - kőr (`Hearts :: Suit`)
  - káró (`Diamonds :: Suit`)
  - treff (`Clubs :: Suit`)
  - pikk (`Spades :: Suit`)

- Definiáld a kártyát repreztáláló `Card` típust. A `Card` típusú értékeket egyetlen `C :: Suit -> Rank -> Card` adatkonstuktorral képezzük, melynek első paramétere a kártya színe, második paramétere a kártya értéke.
- Gondoskodjunk róla, hogy minden általunk definiált típusra legyen értelmezett a `show` függvény és az `==` (illetve a `/=`) művelet!

Párok (1 pont)

Definiáld a `pair :: Card -> Card -> Bool` függvényt, amely eldönti két kártyáról, hogy párok-e azaz, hogy megegyezik-e a két kártya értéke.

```
pair (C Diamonds 10) (C Clubs 10) == True
pair (C Diamonds 14) (C Clubs 2) == False
pair (C Hearts 14) (C Hearts 2) == False
pair (C Spades 2) (C Spades 2) == True
and [ pair (C Clubs i) (C Hearts i) | i <-[1..14]] == True
```

Szín (1 pont)

Definiáld a `flush :: [Card] -> Bool` függvényt, amely kártyák listájáról eldönti, hogy minden kártya azonos színű-e.

Segítség: pontosan akkor ugyanolyan szinű minden kártya, ha a színük megegyezik az első kártya színével. Tekinthetjük úgy, hogy az üres listában minden kártya ugyanolyan színű.

```
flush [C Clubs 12, C Clubs 13, C Clubs 14] == True
flush [C Clubs 9, C Clubs 10, C Hearts 11] == False
flush [C Spades 6, C Diamonds 7, C Spades  8] == False
flush [C Hearts 3, C Spades 4, C Spades 5] == False
flush [C Diamonds 2] == True
flush [] == True
flush (replicate 10 (C Hearts 1)) == True
flush (take 10 (cycle [C Diamonds 13, C Hearts 14])) == False
flush (replicate 9 (C Hearts 11) ++ [C Clubs 12])  == False
```

Első lap keresése (2 pont)

Definiáld a `search :: (Suit -> Rank -> Bool) -> [Card] -> Maybe Card` függvényt, amely egy kártyapakliban megkeresi az első olyan kártyát, melynek adattagjaira teljesül a megadott feltétel. Amennyiben nincs ilyen kártya, az eredmény legyen `Nothing`.

```
search (\s r -> s == Diamonds && r < 10) [C Clubs 9, C Diamonds 14, C Diamonds 3] == Just (C Diamonds 3)
search (\s r -> s == Diamonds && r < 10) [C Clubs 9, C Diamonds 14] == Nothing
search (\s r -> s == Diamonds && r < 10) [C Clubs 9, C Diamonds 3, C Diamonds 14] == Just (C Diamonds 3)
search (\s r -> s == Diamonds && r < 10) [ C Diamonds 3, C Clubs 9,C Diamonds 14] == Just (C Diamonds 3)
search (\s r -> r < 10) (replicate 10 (C Spades 14) ++ [(C Spades 3)]) == Just (C Spades 3)
```

```
search (\s r -> True) [] == Nothing
search (\s r -> False) (replicate 10 (C Clubs 9)) == Nothing
```