Csoportok

Funkcionális programozás
IP-18FUNPEG | 2

Funkcionális programozás
IP-18FUNPEG | 604

Nagybeadandó

Kategória: Beadandók
Elérhető:
Pótolható határidő: ()
Végső határidő: 2024. 05. 17. 23:59 (Lejárt)
Maximum beküldési próbálkozások száma: Korlátlan
Kiírta: Bozó István

Leírás:

Hét csoda párbaj

A feladat során a Hét csoda nevű társasjáték két fős változatát, a **Hét csoda párbaj** (Seven wonders duel) egy (jelentősen) egyszerűsített változatát fogjuk implementálni. Ez egy civilizáció építő játék, ahol a játékosok várost építenek a különféle kártyák segítségével, a párbajt pedig az erősebb város nyeri.

Mivel az implementációban nem lehet grafikai felületünk, ezért csak egy szimulációt fogunk elvégezni, vagyis önmagától fognak zajlani a körök.

Megjegyzések és előfeltételek

Tesztesetek

- A feladatokat a leírásnak megfelelően kell megoldani, de minden feladatnál adottak tesztesetek, amelyek a függvény működésének tesztelését segítik.
- *Tekintve, hogy a tesztesetek - bár odafigyelés mellett íródnak - nem fedik le minden esetben a függvény teljes működését, ezért határozottan javasolt még külön próbálgatni a megoldásokat beadás előtt!*
- A végső teszteléshez használt tesztek halmaza bővebb lehet a megadott teszthalmaznál.

A játékról

- Minden kártyának van egy költsége, azaz amennyiért lehet megvenni (`Cost`). Ez lehet pénz és/vagy nyersanyagok.
  - Ezt az eredeti szabály 5. oldala taglalja. (A beadandó szabályai eltérnek az eredetitől.)

- Egy nyersanyag típus csak egyszer szerepel a `Cost` -ban.
- Minden kártyának van egy hatása, ami típusonként eltér.
- A játékban minden kártya pontosan egyszer található meg. Két kártya hatása lehet ugyanaz, de a nevük mindenképp különbözik.
- A sima kártyák mellett vannak különleges ún. csoda kártyák (lásd bővebben lentebb).
- A játékot két játékos játssza. A játékosok pénzzel, kártyákkal és csodákkal rendelkeznek.
- A játékosnál levő kártyákat (a csoda kártyákat is ideértve) nevezzük városnak vagy a játékos kezében lévő kártyáknak.

A kártya típusokról

A játékban 6-féle (az eredetiben 7) kártya van. A kártyák létesítményeket szimbolizálnak, amelyeknek különféle hatásai vannak.

- Alapanyagok (`Materials`)
  - Nyersanyagokat adnak a játékosoknak.
  - Egy ilyen kártyán egyféle nyersanyag lehet, de akár több darab is.
  - A játékos a már birtokolt kártyáit nem dobja el, amikor "megfizeti" egy épp felvett kártya nyersanyag költségét. A nyersanyag kártyák gyakorlatilag újratermelődnek.

- Civil létesítmények (`Civilian`)
  - Pontszám található rajtuk.

- Tudományos létesítmények (`Scientific`)
  - Pontszám és tudományos szimbólum található rajtuk.
  - A szimbólumokból hatféle van (pontosan 2 darab van mindegyikből).

    - A két egyforma szimbólum extra pontokhoz juttatja a játékost.

- Katonai létesítmények (`Military`)
    - 1-3 darab pajzs található rajtuk.
    - A játék során figyelni kell a pajzsok számára is, hiszen ez a győzelem egyik módja (lásd. később).

- Kereskedelmi létesítmények (`Commercial`)
    - Ezen kártyáknak háromféle hatása lehet, illetve szerepelhet rajtuk pontszám.

        1. Pénzt ad a játékosnak
        2. Bizonyos kártya típus alapján ad pénzt a játékosnak
        3. Egy nyersanyagnak fix árat állít be (lásd. később)

- Céhek (`Guilds`)
    - Pontot adnak a játékosnak egy bizonyos kártya típusért.

A csodákról

- A kártyák egy külön csoportjába tartoznak a csoda kártyák.
- Minden játékosnak 4 darab csoda kártyája van.
- A csodák pajzsokat, pontokat és pénzt adhatnak.
- Egy csodát úgy lehet megépíteni, hogy az értékének megfizetése mellett "aláteszünk" egy kártyát. (Lásd. a `Player` adattípusnál)

A kártyák definiálása

Először a játékban található kártyákat és a hozzájuk szükséges típusokat fogjuk definiálni. A megvalósítás során az összes adattípusra kérjük meg a fordítót, hogy példányosítson `Show` és `Eq` példányt! Amennyiben ezeken felül másra is szükség lenne, azok is megadhatók a későbbiekben.

Másoljuk be az alábbi típusszinonimákat a modulba!

```
type Name = String                  -- kártya vagy játékos neve
type Drachma = Int                  -- a pénz a játékban
type Point = Int                    -- pontszám
type Shield = Int                   -- pajzsok száma
type Cost = ([Product], Drachma)    -- a kártya költsége (a `Product` típus kés
type Table = [[Maybe Card]]         -- a játék tábla (a `Card` típus később les
type IsBuilt = Bool                 -- meg van-e építve (a csodáknál lesz haszn
```

Nyersanyagok

A játékban ötféle nyersanyag van: kő, agyag, fa, papír, üveg.

Definiáljuk a `Product` adattípust, melynek konstruktorai a nyersanyagokat szimbolizálják, ezek: `Clay` , `Wood` , `Stone` , `Glass` , `Papyrus` ! Mindegyiknek legyen egy `Int` paramétere, amely mennyiségét adja meg.

Tudományos szimbólumok

Definiáljuk a tudományos kártyákhoz szükséges szimbólumokat! Az adattípus neve legyen `Symbol` , konstruktorai pedig a következők: `Globe` , `Wheel` , `Sundial` , `Mortar` , `Pendulum` , `Quill` .

Kereskedelmi és céh kártyák hatásai

A kereskedelmi és céh kártyáknak több különböző hatása lehet. Definiáljunk hozzá egy `Effect` adattípust! A konstruktorai az alábbiak legyenek:

Ilyen kártyán szerepelhet	Konstruktor	Paraméterek	Hatás	Megjegyzés

Ilyen kártyán szerepelhet	Konstruktor	Paraméterek	Hatás	Megjegyzés
`Commercial`	`Price`	nyersanyag (`Product`)	Fix árat határoz meg egy nyersanyag típusnak. Minden nyersanyaghoz 1 db van. Ha ilyennel rendelkezik a játékos és kellene vennie az adott nyersanyagból, akkor csak 1 drachmát kell fizetnie	A hatása végig érvényesül
`Commercial`	`Money`	pénz (`Drachma`)	A paraméterben meghatározott mennyiségű pénzt ad a játékosnak	Azonnal érvényesül a hatása, amikor a játékoshoz kerül
`Commercial`	`MoneyByCard`	kártya (`Card`), pénz (`Drachma`)	Pénzt ad a játékosnak a megadott kártyatípus után	Azonnal érvényesül a hatása, amikor a játékoshoz kerül
`Guild`	`PointsByCard`	Első paramétere egy `Either` típusú érték, aminek a két típusparamétere `Card` és `WonderCard` , a második a pontszám (`Point`), ami az adott típusú kártyákért járt	Pontot ad a játékosnak a megadott kártyatípus után	A játék végén érvényesül a pontozásnál

(A `Card` és a `WonderCard` típus a későbbiekben kerül bevezetésre.)

A kártyák

Definiáljuk a kártyák adattípusát `Card` néven. A játékban 6-féle (az eredetiben 7) kártya van. A kártyák létesítményeket szimbolizálnak, amelyeknek különféle hatásai vannak. A kártyák megépítéséhez nyersanyagokra és/vagy pénzre van szükség (`Cost`). A konstruktorok paramétereit a táblázat tartalmazza.

Konstruktor	Megnevezés	Paraméterek
`Materials`	Alapanyagok	név (`Name`), költség (`Cost`), nyersanyag (`Product`)
`Civilian`	Civil létesítmények	név (`Name`), költség (`Cost`), pontszám (`Point`)
`Scientific`	Tudományos létesítmények	név (`Name`), költség (`Cost`), pontszám (`Point`), tudományos szimbólum (`Symbol`)
`Military`	Katonai létesítmények	név (`Name`), költség (`Cost`), pajzsok (`Shield`)

Konstruktor	Megnevezés	Paraméterek
`Commercial`	Kereskedelmi létesítmények	név (`Name`), költség (`Cost`), pontszám (`Point`), hatás (`Effect`)
`Guilds`	Céhek	név (`Name`), költség (`Cost`), hatás (`Effect`)

(Az `Effect` típus korábban került bevezetésre.)

Csoda kártyák

A csodákat reprezentáló kártyák speciális szerepet töltenek be a játékban, ezért külön adattípusként kerülnek megadásra. Definiáljuk a `WonderCard` adattípust, amelynek egy `W` nevű konstruktora legyen. Ez egy csodát reprezentál. A konstruktor tárolja a csoda nevét (`Name`), a megépítésének költségét (`Cost`), a csoda kártya pont értékét (`Point`), a pajzsok számát (`Shield`) és az érte járó pénzt (`Drachma`).

Játékosok

A játékosokat is egy külön adattípussal adjuk meg, ennek `Player` lesz a neve. Egyetlen konstruktora `P` legyen, paraméterei tartalmazzák a játékos nevét (`Name`), a kártyáit (`Card`) egy listában, a csoda kártyáit `[(WonderCard, IsBuilt)]` formában és a pénzét (`Drachma`).

Az összes kártya

Az adattípusok definiálása után másoljuk az alábbi minta kártyákat is a modulba!

```
blankMaterialsCard = (Materials "" ([], 0) (Stone 0))
blankMilitaryCard = (Military "" ([], 0) 0)
blankCivilianCard = (Civilian "" ([], 0) 0)
blankScientificCard = (Scientific "" ([], 0) 0 (Quill))
blankCommercialCard = (Commercial "" ([], 0) 0 (Price (Stone 0)))
blankWonderCard = (W "" ([], 0) 0 0 0)
```

Az alábbiakat nem kötelező bemásolni, de ezzel tesztelhetjük a kártyákat és a csoda kártyákat. (Ez a játékban előforduló összes kártya.)

```
allCards :: [Card]
allCards = [
   (Materials "Lumber yard" ([], 0) (Wood 1)),
   (Materials "Logging camp" ([], 1) (Wood 1)),
   (Materials "Clay pool" ([], 0) (Clay 1)),
   (Materials "Clay pit" ([], 1) (Clay 1)),
   (Materials "Quarry" ([], 0) (Stone 1)),
   (Materials "Stone pit" ([], 1) (Stone 1)),
   (Materials "Sawmill" ([], 2) (Wood 2)),
   (Materials "Brickyard" ([], 2) (Clay 2)),
   (Materials "Shelf quarry" ([], 2) (Stone 2)),
   (Materials "Glassworks" ([], 1) (Glass 1)),
   (Materials "Glass blowers" ([], 0) (Glass 1)),
   (Materials "Press" ([], 1) (Papyrus 1)),
   (Materials "Dying room" ([], 0) (Papyrus 1)),

   (Civilian "Theater" ([], 0) 3),
   (Civilian "Altar" ([], 0) 3),
   (Civilian "Baths" ([(Stone 1)], 0) 3),
   (Civilian "Tribunal" ([(Wood 2), (Glass 1)], 0) 5),
   (Civilian "Statue" ([(Clay 2)], 0) 4),
   (Civilian "Temple" ([(Wood 1), (Papyrus 1)], 0) 4),
   (Civilian "Aqueduct" ([(Stone 3)], 0) 5),
   (Civilian "Rostrum" ([(Stone 1), (Wood 1)], 0) 4),
   (Civilian "Palace" ([(Clay 1), (Stone 1), (Wood 1), (Glass 2)], 0) 7),
   (Civilian "Town hall" ([(Stone 3), (Wood 2)], 0) 7),
   (Civilian "Obelisk" ([(Stone 2), (Glass 1)], 0) 5),
   (Civilian "Gardens" ([(Clay 2), (Wood 2)], 0) 6),
   (Civilian "Pantheon" ([(Clay 1), (Wood 1), (Papyrus 2)], 0) 6),
   (Civilian "Senate" ([(Clay 2), (Stone 1), (Papyrus 1)], 0) 5),

   (Scientific "Workshop" ([(Papyrus 1)], 0) 1 (Pendulum)),
   (Scientific "Apothecary" ([(Glass 1)], 0) 1 (Wheel)),
   (Scientific "Academy" ([(Stone 1), (Wood 1), (Glass 2)], 0) 3 (Sundial)),
   (Scientific "Study" ([(Papyrus 1), (Glass 1), (Wood 2)], 0) 3 (Sundial)),
   (Scientific "Scriptorium" ([], 2) 0 (Quill)),
   (Scientific "Library" ([(Stone 1), (Wood 1), (Glass 1)], 0) 2 (Quill)),
   (Scientific "Pharmacist" ([], 2) 0 (Mortar)),
   (Scientific "Dispensary" ([(Papyrus 1)], 0) 2 (Mortar)),
   (Scientific "School" ([(Papyrus 1)], 0) 1 (Wheel)),
   (Scientific "University" ([(Papyrus 1)], 0) 2 (Globe)),
   (Scientific "Laboratory" ([(Papyrus 1)], 0) 1 (Pendulum)),
   (Scientific "Observatory" ([(Papyrus 1)], 0) 2 (Globe)),

   (Military "Guard tower" ([], 0) 1),
   (Military "Walls" ([(Stone 2)], 0) 2),
   (Military "Arsenal" ([(Clay 3), (Wood 2)], 0) 3),
   (Military "Courthouse" ([], 8) 3),
   (Military "Stable" ([(Wood 1)], 0) 1),
   (Military "Horse breeders" ([(Wood 1), (Clay 1)], 0) 1),
   (Military "Garrison" ([(Clay 1)], 0) 1),
   (Military "Barracks" ([], 3) 1),
   (Military "Palisade" ([], 2) 1),
   (Military "Fortifications" ([(Stone 2), (Clay 1), (Papyrus 1)], 0) 2),
   (Military "Archery range" ([(Stone 1), (Wood 1), (Papyrus 1)], 0) 2),
   (Military "Siege workshop" ([(Wood 3), (Glass 1)], 0) 2),
   (Military "Parade ground" ([(Clay 2), (Glass 1)], 0) 2),
   (Military "Circus" ([(Stone 2), (Clay 2)], 0) 2),

   (Commercial "Stone reserve" ([], 3) 0 (Price (Stone 0))),
   (Commercial "Clay reserve" ([], 3) 0 (Price (Clay 0))),
   (Commercial "Wood reserve" ([], 3) 0 (Price (Wood 0))),
   (Commercial "Customs house" ([], 3) 0 (Price (Papyrus 0))),
   (Commercial "Brewery" ([], 3) 0 (Price (Glass 0))),
   (Commercial "Tavern" ([], 0) 0 (Money 6)),
   (Commercial "Forum" ([], 0) 0 (Money 4)),
   (Commercial "Caravansery" ([(Glass 1), (Papyrus 1)], 0) 0 (Money 8)),
   (Commercial "Arena" ([(Stone 1), (Wood 1)], 0) 0 (Money 7)),
   (Commercial "Chamber of commerce" ([(Papyrus 2)], 0) 3 (MoneyByCard blankMat
   (Commercial "Port" ([(Wood 1), (Glass 1), (Papyrus 1)], 0) 3 (MoneyByCard bl
   (Commercial "Armory" ([(Stone 2), (Glass 1)], 0) 3 (MoneyByCard blankMilitar
   (Commercial "Lighthouse" ([(Clay 2), (Glass 1)], 0) 3 (MoneyByCard blankCivi

   (Guilds "Shipowners guild" ([(Clay 1), (Stone 1), (Glass 1), (Papyrus 1)], 0
   (Guilds "Tacticians guild" ([(Stone 2), (Clay 1), (Papyrus 1)], 0) (PointsBy
   (Guilds "Magistrates guild" ([(Wood 2), (Clay 1), (Papyrus 1)], 0) (PointsBy
   (Guilds "Scientists guild" ([(Wood 2), (Clay 2)], 0) (PointsByCard (Left bla
   (Guilds "Merchants guild" ([(Clay 1), (Wood 1), (Glass 1), (Papyrus 1)], 0)
   (Guilds "Builders guild" ([(Stone 2), (Clay 1), (Wood 1), (Glass 1)], 0) (Po
   (Guilds "Moneylenders guild" ([(Stone 2), (Wood 2)], 0) (PointsByCard (Right

allWonders :: [WonderCard]
allWonders = [
   (W "The Appian Way" ([(Papyrus 1), (Clay 2), (Stone 2)], 0) 3 0 3),
   (W "Circus Maximus" ([(Glass 1), (Wood 1), (Stone 2)], 0) 3 1 0),
   (W "The Colossus" ([(Glass 1), (Clay 3)], 0) 3 2 0),
   (W "The Great Library" ([(Papyrus 1), (Glass 1), (Wood 3)], 0) 4 0 0),
   (W "The Great Lighthouse" ([(Papyrus 2), (Stone 1), (Wood 1)], 0) 4 0 0),
```

```
    (W "The Hanging Gardens" ([(Papyrus 1), (Glass 1), (Wood 2)], 0) 3 0 6),
    (W "The Mausoleum" ([(Papyrus 1), (Glass 2), (Clay 2)], 0) 2 0 0),
    (W "Piraeus" ([(Clay 1), (Stone 1), (Wood 2)], 0) 2 0 0),
    (W "The Pyramids" ([(Papyrus 1), (Stone 3)], 0) 9 0 0),
    (W "The Sphinx" ([(Glass 2), (Clay 1), (Stone 1)], 0) 6 0 0),
    (W "The Statue of Zeus" ([(Papyrus 2), (Clay 1), (Wood 1), (Stone 1)], 0) 3
    (W "The Temple of Artemis" ([(Papyrus 1), (Glass 1), (Stone 1), (Wood 1)], (
```

Kötelező feladatok

Ezen feladatok megoldása mindenkinek kötelező, szükséges feltétele a jegyszerzésnek.

Alap pontok összeszámolása

A játékban 2 (az eredetiben 3) győzelmi típus van, menet közben katonai fölényre tesz szert valamelyik játékos, vagy a játék végén az erősebb város arat győzelmet. Ez utóbbi meghatározása lesz a feladatunk ebben a részben.

A játék végén összesítésre kerülnek a megszerzett pontok, a versenyt az a játékos nyeri, akinek a városa több pontot ér. Definiáljuk a `countBasicPoints` függvényt, amely visszaadja, hogy a paraméterül kapott játékos hány pontot gyűjtött a játék során.

Az alábbiakat kell figyelembe venni:

- Pontok a kártyákon:
  - A civil létesítmények és a tudományos kártyák adhatnak pontokat.
  - A csoda kártyák adhatnak pontokat. (Csak akkor számítanak, ha meg vannak építve!)
- Pajzsokért járó pont:
  - Számoljuk össze a pajzsokat, amelyek a játékos katonai és a csoda kártyáin szerepelnek. (A csoda kártyák csak akkor számítanak, ha meg vannak építve!)
  - A pajzsok az alábbiak szerint adnak pontot:
    - 1 <= pajzsok < 3 : 2 pont
    - 3 <= pajzsok < 6 : 5 pont
    - 6 <= pajzsok : 10 pont
    - Egyéb esetben nem jár pont
- Minden céh kártyaért 1-1 pontot számoljunk.
- A játékos a pénzéért 3 drachmánként egy 1 pontot kapjon.

```
countBasicPoints :: Player -> Point
```

Tesztek

```
countBasicPoints (P "" [] [] 0) == 0

countBasicPoints (P "Viktor" [] [] 9) == 3

countBasicPoints (P "Vivi" [] [] 18) == 6

countBasicPoints (P "Hello Harold" [] [((W "The Temple of Artemis" ([(Papyrus 1

countBasicPoints (P "Hello Harold" [] [((W "The Temple of Artemis" ([(Papyrus 1

countBasicPoints (P "Hello Harold" [] [((W "The Temple of Artemis" ([(Papyrus 1

countBasicPoints (P "Hello Harold" [] [((W "The Great Library" ([(Papyrus 1), (

countBasicPoints (P "Aba" [(Commercial "Customs house" ([], 3) 0 (Price (Papyru

countBasicPoints (P "Tina" [(Military "Guard tower" ([], 0) 1),(Military "Walls

countBasicPoints (P "Hello Harold" [] [((W "The Temple of Artemis" ([(Papyrus 1

countBasicPoints (P "Jancsi" [(Civilian "Statue" ([(Clay 2)], 0) 4),(Military "

countBasicPoints (P "André" [(Commercial "Brewery" ([], 3) 0 (Price (Glass 0))]

countBasicPoints (P "Geri" [(Military "Horse breeders" ([(Wood 1), (Clay 1)], (
```

Katonai győzelem

Jelen feladatban a másik győzelmi típust, a katonai fölényt fogjuk megvizsgálni. Amennyiben a játék közben az egyik játékos pajzsainak száma 9-cel nagyobb lesz, mint a másiké, akkor vége a játéknak és a több pajzzsal rendelkező játékos győzött. Számoljuk össze a pajzsokat, amelyek a játékos által birtokolt katonai és a csoda kártyákon szerepelnek!

Definiáljuk a `militaryVictory` nevű függvényt, amely ellenőrzi, hogy a megadott állás szerint fennáll-e a katonai győzelem valamelyik játékosnál. A függvény két játékost kap paraméterül, amelyeket felhasználva összeveti a két játékos pajzsainak számát. A függvény a nyertes játékost adja vissza `Maybe Player` -ként, ha történt katonai győzelem. Ha nem történt, akkor `Nothing` -ot adjon vissza.

```
militaryVictory :: Player -> Player -> Maybe Player
```

Tesztek

```
militaryVictory (P "Viktor" [] [] 9) (P "Vivi" [] [] 18) == Nothing

militaryVictory (P "Viktor" [(Military "Guard tower" ([], 0) 1),(Military "Wal

militaryVictory (P "Viktor" [(Civilian "Gardens" ([(Clay 2), (Wood 2)], 0) 6),

militaryVictory (P "Viktor" [(Civilian "Gardens" ([(Clay 2), (Wood 2)], 0) 6),

militaryVictory (P "Anna" [(Commercial "Armory" ([(Stone 2), (Glass 1)], 0) 3

militaryVictory (P "Anna" [(Commercial "Armory" ([(Stone 2), (Glass 1)], 0) 3
```

Kártya vagy csoda ára

A játékosok a játék folyamán kártyákat vásárolnak, hogy különböző előnyökre tegyenek szert. Azonban mindennek megvan az ára (`Cost`), az egyes kártyák nyersanyagokba és/vagy pénzbe kerülhetnek.

A nyersanyagokat a játékos saját (már felvett) kártyáival tudja megtéríteni, de ha nem rendelkezik megfelelő mennyiséggel, úgy azt pénzért is megvásárolhatja.

Vegyük figyelembe az alábbiakat:

- Egy egységnyi nyersanyagot 3 drachmáért lehet megvásárolni.
- A játékos kereskedelmi (`Commercial`) kártyái között lehetnek olyanok, amelyek hatása, hogy fix árat állít be egy nyersanyagnak. A fix ár minden ilyen kártyánál 1 drachmát jelent! (Ez az ár az összes adott típusú nyersanyagra vonatkozik.)
- A kártyán lehet egyszerre pénz és nyersanyag költség is. Ekkor mindkettőt meg kell fizetni.
- Azon nyersanyagok, amelyeket a játékos nyersanyag kártyái fedeznek azok természetesen 0 drachmának számítanak. (A felhasznált nyersanyag kártyák a játékos kezében maradnak!)

Adottak egy játékos kártyái (ezek között lehetnek nyersanyagkártyák is (`Materials`)). A feladatunk az, hogy ezek ismeretében, meghatározzuk azt, hogy egy másik kártya költsége mennyi pénzbe kerülne, ha azt meg szeretnénk vásárolni.

Definiáljuk a `costInMoney` függvényt! Első paramétere egy `Cost` , ami a megvásárolni kívánt kártya vagy csoda költsége. A második egy kártya lista, ami a játékos saját kártyáit jelenti.

```
costInMoney :: Cost -> [Card] -> Drachma
```

Tesztek

```
costInMoney ([], 0) [] == 0

costInMoney ([], 3) [] == 3

costInMoney ([(Wood 1)], 0) [(Materials "Logging camp" ([], 1) (Wood 1))] == 0

costInMoney ([(Wood 1)], 0) [(Materials "Sawmill" ([], 2) (Wood 2))] == 0

costInMoney ([], 0) [(Military "Circus" ([(Stone 2), (Clay 2)], 0) 2),(Civilia

costInMoney ([(Clay 1), (Stone 1), (Glass 1), (Papyrus 1)], 0) [(Materials "Cl

costInMoney ([(Clay 1), (Stone 1), (Glass 1), (Papyrus 1)], 5) [(Materials "Cl

costInMoney ([(Wood 1), (Clay 1)], 0) [(Commercial "Clay reserve" ([], 3) 0 (P

costInMoney ([(Clay 3), (Stone 1), (Wood 1), (Glass 1), (Papyrus 1)], 1) [(Com

costInMoney ([(Clay 1), (Stone 1), (Wood 1), (Glass 1), (Papyrus 1)], 0) [] ==

costInMoney ([(Clay 1), (Stone 1), (Wood 1), (Glass 1), (Papyrus 1)], 0) [(Mat

costInMoney ([(Clay 1), (Stone 1), (Wood 2), (Glass 1), (Papyrus 1)], 1) [(Mat

costInMoney ([(Clay 3), (Stone 1), (Glass 1)], 0) [(Commercial "Wood reserve"

costInMoney ([(Clay 1), (Wood 1), (Papyrus 2)], 2) [(Military "Fortifications"

costInMoney ([(Clay 1), (Stone 1), (Wood 1), (Glass 1), (Papyrus 1)], 0) [(Com

costInMoney ([(Clay 1), (Stone 1), (Glass 1)], 1) [(Materials "Brickyard" ([],
```

Példa - Kártya költség

A játékos szeretne venni egy kártyát, amelynek költsége:

- 1 kő
- 2 agyag
- 1 fa
- 1 drachma

A játékos kezében van:

- 2 kő
- 1 üveg
- fix ár kártya az agyaghoz

Így alakul az összeg:

- A kőért nem kell fizetnie, mert van a kezében: 0 drachma
- Agyaggal nem rendelkezik, de van fix ár kártyája hozzá: 2 * 1 drachma
- Fája nincs, sem fix ár kártyája hozzá: 3 drachma
- A kártyának van pénz értéke is: 1 drachma
- Az összeg: 0 + (2 * 1) + 3 + 1 = 6

Céh kártyák hatása

A céh kártyák a játék végén adnak pontokat a játékosoknak. Minden céh kártyán szerepel egy minta kártya vagy csoda, ami alapján ad megadott darab pontot kártyánként. Mindkét játékos kártyáit vizsgálni kell, mivel annyi darab kártyáért jár a pont, amennyi abban a városban van, ahol a legtöbb adott típusú kártya van.

Definiáljuk a `guildPoints` függvényt, amely megadja egy játékos céhekből származó pontszámát. A függvény két játékost kapjon paraméterül és a pontszámot adja vissza. Az első paraméterként kapott játékos céh kártyáit kell számolni!

```
guildPoints :: Player -> Player -> Point
```

Tesztek

```
guildPoints (P "Zsófi" [] [] 0) (P "Lili" [] [] 0) == 0

guildPoints (P "Zsófi" [] [((W "The Statue of Zeus" ([(Papyrus 2), (Clay 1), (

guildPoints (P "Zsófi" [(Guilds "Tacticians guild" ([(Stone 2), (Clay 1), (Papy

guildPoints (P "Zsófi" [(Military "Palisade" ([], 2) 1),(Military "Fortificati

guildPoints (P "Bence" [(Guilds "Scientists guild" ([(Wood 2), (Clay 2)], 0) (

guildPoints (P "Dóri" [(Guilds "Merchants guild" ([(Clay 1), (Wood 1), (Glass 1

guildPoints (P "Réka" [(Guilds "Builders guild" ([(Stone 2), (Clay 1), (Wood 1]

guildPoints (P "István" [(Guilds "Moneylenders guild" ([(Stone 2), (Wood 2)], 0
```

Példa - Céh

Az Annánál lévő céh kártya a tudományos kártyákat számolja. Annának 3 tudományos kártyája van, Botondnak pedig 5 db. Ekkor Annának 5 pont jár, hiába vannak azok Botond városában.

Tudományos kártyák extra pontjai

A tudományos kártyákon összesen 6-féle szimbólum található, mindegyikből 2-2. Ezen kártyákért pontok is járnak. Ha egy játékosnak van két egyforma szimbólumú tudományos kártyája, akkor kapja meg az értük járó pontok dupláját extra pontként. Definiáljuk a `scientificPlusPoints` függvényt, amely egy játékost kap paraméterül és ezt az extra pontszámot adja vissza.

```
scientificPlusPoints :: Player -> Point
```

Tesztek

```
scientificPlusPoints (P "Ibolya" [] [] 0) == 0

scientificPlusPoints (P "Ilona" [] [] 5) == 0

scientificPlusPoints (P "Márton" [(Commercial "Port" ([(Wood 1), (Glass 1), (Pa

scientificPlusPoints (P "Levente" [(Materials "Press" ([], 1) (Papyrus 1)),(Ma

scientificPlusPoints (P "Áron" [(Civilian "Pantheon" ([(Clay 1), (Wood 1), (Pap

scientificPlusPoints (P "Dániel" [(Scientific "Scriptorium" ([], 2) 0 (Quill))

scientificPlusPoints (P "Tamás" [(Scientific "Apothecary" ([(Glass 1)], 0) 1 (W
```

Extra feladatok

A feladatok megoldása opcionális, azaz nem kötelező. Ebből a csoportból bármelyik feladat megoldható, a megszerezett pont hozzáadódik a vizsga eredményéhez (maximum 5 pont erejéig).

A továbbiakban a `game` függvényt vagyis a teljes játékot szimuláló függvényt készítjük elő.

Tábla (2 pont)

Ebben játékban a kártyák alkotják a táblát, amelyek egy piramis alakzatban vannak felrakva. Alulról nézve 6, 5, 4, 3, 2 db kártya található az egyes sorokban.

```
 (4. sor)       0 1
 (3. sor)      0 1 2
 (2. sor)     0 1 2 3
 (1. sor)    0 1 2 3 4
 (0. sor)   0 1 2 3 4 5

[ [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4], [0, 1, 2, 3], [0, 1, 2], [0, 1] ]
```

Az eredeti szabály leírás 20. oldalán találunk erről ábrázolást. (Az 1. kor ábráját kell nézni és mi nem fordítjuk le a kártyákat.)

Egy kártyát akkor lehet felvenni, ha nincs rajta másik kártya. Tehát a 0. sor kivételével minden kártyát két másik kártya takar kezdetben. A táblát egy `Table` típus definiálja, ahol a 0. lista a 0. (azaz a legalsó) sort reprezentálja. Ahhoz, hogy a sorokat összefogó lista szerkezete ne változzon meg a táblából kivett kártyák helyén `Nothing` szerepel.

Definiáljuk az `isCardFree` függvényt, amely visszaadja, hogy az adott kártyát fel lehet-e venni. Amennyiben a kártya nem szerepel a táblában, akkor ez mindenképp hamis legyen. A függvény első paramétere a felvenni kívánt kártya, a második a tábla legyen. A megoldáshoz érdemes használni a fentebbi indexelést.

```
isCardFree :: Card -> Table -> Bool
```

Tesztek

```
not $ isCardFree (Scientific "Dispensary" ([(Papyrus 1)], 0) 2 (Mortar)) [[]]

not $ isCardFree (Scientific "Laboratory" ([(Papyrus 1)], 0) 1 (Pendulum)) [[J

not $ isCardFree (Military "Barracks" ([], 3) 1) [[Just (Military "Courthouse"

isCardFree (Military "Courthouse" ([], 8) 3) [[Just (Military "Courthouse" ([]

isCardFree (Materials "Glass blowers" ([], 0) (Glass 1)) [[Nothing,Nothing,Not

not $ isCardFree (Scientific "Library" ([(Stone 1), (Wood 1), (Glass 1)], 0) 2

not $ isCardFree (Scientific "Study" ([(Papyrus 1), (Glass 1), (Wood 2)], 0) 3

isCardFree (Commercial "Clay reserve" ([], 3) 0 (Price (Clay 0))) [[Nothing,No

isCardFree (Materials "Shelf quarry" ([], 2) (Stone 2)) [[Nothing,Nothing,Noth

isCardFree (Materials "Quarry" ([], 0) (Stone 1)) [[Nothing,Nothing,Nothing,No

isCardFree (Military "Courthouse" ([], 8) 3) [[Nothing,Nothing,Nothing,Nothing

isCardFree (Civilian "Rostrum" ([(Stone 1), (Wood 1)], 0) 4) [[Nothing,Nothing

not $ isCardFree (Military "Archery range" ([(Stone 1), (Wood 1), (Papyrus 1)]
```

Kártya felvétele (1 pont)

<!-- Épül az előzőre -->
Definiáljuk a `cardToNothing` függvényt, amely kiveszi az első paraméterként kapott kártyát a táblából. A tábla szerkezete ne változzon, a kártya helyére `Nothing` kerüljön.

```
cardToNothing :: Card -> Table -> Table
```

Tesztek

```
cardToNothing (Scientific "Dispensary" ([(Papyrus 1)], 0) 2 (Mortar)) [[]] ==

cardToNothing (Military "Courthouse" ([], 8) 3) [[Just (Military "Courthouse"

cardToNothing (Materials "Glass blowers" ([], 0) (Glass 1)) [[Nothing,Nothing,

cardToNothing (Commercial "Clay reserve" ([], 3) 0 (Price (Clay 0))) [[Nothing

cardToNothing (Materials "Shelf quarry" ([], 2) (Stone 2)) [[Nothing,Nothing,N

cardToNothing (Materials "Quarry" ([], 0) (Stone 1)) [[Nothing,Nothing,Nothing

cardToNothing (Military "Courthouse" ([], 8) 3) [[Nothing,Nothing,Nothing,Noth

cardToNothing (Scientific "Laboratory" ([(Papyrus 1)], 0) 1 (Pendulum)) [[Just

cardToNothing (Military "Archery range" ([(Stone 1), (Wood 1), (Papyrus 1)], 0
```

Megvásárolható-e (2 pont)

A kártyák, illetve csodák árát már egy korábbi feladatban meghatároztuk. Most állapítsuk meg, hogy ez alapján meg tudja-e vásárolni azt a kártyát vagy csodát. Azt nem kell ellenőrizni, hogy nincs-e már a játékosnál, mivel egy kártyából pontosan egy van. Valamint azt sem, hogy ezek a táblán vannak-e. A csodáknál viszont ellenőrizni

kell, hogy az adott csoda kártya a játékosnál van-e. Ellenőrizzük azt is, hogy nincs-e még megépítve. Ha nincs a játékosnál vagy már meg van építve, akkor adjunk vissza hamisat!

Definiáljuk a `canBuyCard` és a `canBuyWonder` függvényeket. Az előbbi első paramétere egy kártya, az utóbbié egy csoda legyen. A második paraméter mindkettőnél a játékos.

```
canBuyCard :: Card -> Player -> Bool

canBuyWonder :: WonderCard -> Player -> Bool
```

Tesztek

```
canBuyCard (Materials "Lumber yard" ([], 0) (Wood 1)) (P "Emese" [] [] 7)

canBuyCard (Materials "Lumber yard" ([], 0) (Wood 1)) (P "Orsolya" [] [((W "The

not $ canBuyCard (Scientific "Laboratory" ([(Papyrus 1)], 0) 1 (Pendulum)) (P

canBuyCard (Materials "Lumber yard" ([], 0) (Wood 1)) (P "Márk" [] [((W "The A

canBuyCard (Scientific "Laboratory" ([(Papyrus 1)], 0) 1 (Pendulum)) (P "Eszte

not $ canBuyCard (Commercial "Customs house" ([], 3) 0 (Price (Papyrus 0))) (P

canBuyCard (Civilian "Senate" ([(Clay 2), (Stone 1), (Papyrus 1)], 0) 5) (P "N

not $ canBuyCard (Civilian "Senate" ([(Clay 2), (Stone 1), (Papyrus 1)], 0) 5)

canBuyCard (Scientific "Study" ([(Papyrus 1), (Glass 1), (Wood 2)], 0) 3 (Sund

not $ canBuyCard (Guilds "Tacticians guild" ([(Stone 2), (Clay 1), (Papyrus 1)
```

```
not $ canBuyWonder (W "The Great Library" ([(Papyrus 1), (Glass 1), (Wood 3)],

canBuyWonder (W "The Great Library" ([(Papyrus 1), (Glass 1), (Wood 3)], 0) 4 (

not $ canBuyWonder (W "The Great Library" ([(Papyrus 1), (Glass 1), (Wood 3)],

canBuyWonder (W "The Great Library" ([(Papyrus 1), (Glass 1), (Wood 3)], 0) 4 (

canBuyWonder (W "The Hanging Gardens" ([(Papyrus 1), (Glass 1), (Wood 2)], 0) 1

not $ canBuyWonder (W "The Hanging Gardens" ([(Papyrus 1), (Glass 1), (Wood 2)

not $ canBuyWonder (W "The Pyramids" ([(Papyrus 1), (Stone 3)], 0) 9 0 0) (P "(
```

Akciók (4 pont)

Ez egy összetettebb részfeladat, ahol az akciókat kell reprezentálnunk és elkészítenünk. Egy játékos akciói a következők lehetnek:

1. Eldobja a kártyát
2. Megépíti a kártyát
3. Megépít egy csodát

Adattípus

Definiáljunk egy adattípust `Action` néven, amely a fenti akciókat reprezentálja! Első konstruktora a `DropCard`, ami egy kártyát kapjon paraméterül. A második a `BuildWonder`, amely egy kártyát és egy csodát kapjon paraméterként. A harmadik pedig a `BuildCard`, ami egy kártyát kapjon paraméterül. Kérjük meg a fordítót, hogy példányosítson rá `Show` instance-t!

Kártya eldobása

A kártya eldobása pénzt ad a játékosnak. Ezen összeg meghatározásához számoljuk össze a játékos kereskedelmi kártyáit és ahhoz adjunk hozzá kettőt. Ha a játékosnak nincs ilyen fajta kártyája akkor is jár a kettő drachma. Definiáljuk a `dropCard` függvényt, amely az első paraméterként adott játékos pénzéhez hozzáadja a kártyáért járó összeget!

```
dropCard :: Player -> Card -> Player
```

Kártya megépítése

Egy kártyát úgy építünk meg, hogy a költségét megfizetjük. Definiáljuk a `buildCard` függvényt, amely a paraméterként kapott kártyát hozzáadja a játékoskártyáihoz és levonja az építési költséget. Ne feledjük, hogy van két olyan kereskedelmi kártya, amely pénzt ad (`Money` és `MoneyByCard`). Ezt az összeget adjuk a játékos pénzéhez!

- 1. paraméter: a játékos
- 2. paraméter: a kártya

```
buildCard :: Player -> Card -> Player
```

Csoda megépítése

Egy csodát úgy építünk meg, hogy a költségének megfizetése mellett egy kártyát alá teszünk. A csodák már a játékos kezében vannak, így ellenőrizni kell, hogy a paraméterként kapott csoda tényleg a játékosnál van-e. Ha nincs, akkor a lépés érvénytelen és a játékost változtatás nélkül adjuk vissza. Ha igen, akkor a játékosnál cseréljük a csoda melletti `False` -t `True` -ra. Definiáljuk ezt a függvényt `buildWonder` néven. Ne feledjük, hogy egy csoda adhat pénzt a játékosnak. Ezt az összeget adjuk a játékos pénzéhez!

- 1. paraméter: a játékos
- 2. paraméter: a megvenni kívánt csoda

```
buildWonder :: Player -> WonderCard -> Player
```

Tesztek - Akciók

```
dropCard (P "Alma" [] [] 0) (Civilian "Altar" ([], 0) 3) == (P "Alma" [] [] 2)
dropCard (P "Zsolt" [] [((W "The Mausoleum" ([(Papyrus 1), (Glass 2), (Clay 2)]
not $ dropCard (P "Norbert" [] [((W "The Mausoleum" ([(Papyrus 1), (Glass 2), (
dropCard (P "Boglárka" [(Commercial "Arena" ([(Stone 1), (Wood 1)], 0) 0 (Money
dropCard (P "Melinda" [(Commercial "Lighthouse" ([(Clay 2), (Glass 1)], 0) 3 (M
dropCard (P "Piroska" [(Civilian "Altar" ([], 0) 3),(Materials "Logging camp" (
dropCard (P "István" [(Commercial "Arena" ([(Stone 1), (Wood 1)], 0) 0 (Money 1
```

```
buildCard (P "Alma" [] [] 0) (Civilian "Altar" ([], 0) 3) == (P "Alma" [Civilia
buildCard (P "Marcell" [(Materials "Sawmill" ([], 2) (Wood 2)),(Materials "Bric
buildCard (P "Kristóf" [] [((W "The Sphinx" ([(Glass 2), (Clay 1), (Stone 1)],
buildCard (P "Kinga" [] [((W "The Statue of Zeus" ([(Papyrus 2), (Clay 1), (Woo
buildCard (P "Gergő" [(Military "Parade ground" ([(Clay 2), (Glass 1)], 0) 2),(
buildCard (P "Zsolt" [] [((W "The Mausoleum" ([(Papyrus 1), (Glass 2), (Clay 2]
```

```
buildWonder (P "Ella" [] [] 11) (W "The Great Lighthouse" ([(Papyrus 2), (Stone
buildWonder (P "Renáta" [] [((W "The Great Lighthouse" ([(Papyrus 2), (Stone 1]
buildWonder (P "Lóránt" [] [((W "The Colossus" ([(Glass 1), (Clay 3)], 0) 3 2 (
buildWonder (P "Lilla" [(Materials "Stone pit" ([], 1) (Stone 1)),(Materials "S
buildWonder (P "Tibor" [(Commercial "Stone reserve" ([], 3) 0 (Price (Stone 0)]
buildWonder (P "Szilvia" [(Materials "Brickyard" ([], 2) (Clay 2)),(Commercial
buildWonder (P "Gábriel" [(Scientific "Pharmacist" ([], 2) 0 (Mortar)),(Materia
```

A játék lejátszása (5 pont)

(Ha eddig a feladatig eljutottál, akkor veregesd meg a válladat! :))

Definiáljuk a `game` függvényt, amely lejátssza a játékot!

- 1. paraméter: egyik játékos
- 2. paraméter: másik játékos
- 3. paraméter: az akciók listája
- 4. paraméter: a táblák listája

A táblákból 3-nak kell lennie, mivel a játék ennyi körből áll. A kártyák minden körben ugyanolyan alakzatban vannak felrakva. A játékosok egymás után jönnek. Ismétlés kizárólag akkor van, ha az adott akció nem hajtható végére.(Pl. az 1. akciót az első játékos hajtja végre, a 2.-at a második, a 3.-at az első és így tovább, ha nincs hiba.)

Az akciókhoz az alábbiakat kell figyelembe venni:

1. Eldobja a kártyát
   - Felveheti-e ezt a kártyát?
     - Ha nem, akkor a játékos lépjen a következő akcióra
   - Az eldobásért pénzt kap
2. Megépíti a kártyát
   - Felveheti-e a kártyát?
     - Ha nem, akkor a játékos lépjen a következő akcióra
   - Van-e az építéshez elég nyersanyaga és pénze?
     - Ha nem, akkor eldobja a kártyát
   - Megveszi a kártyát, a kezébe kerül és használhatja
3. Megépít egy csodát
   - Felveheti-e a kártyát?
     - Ha nem, akkor a játékos lépjen a következő akcióra
   - Van-e elég nyersanyaga és pénze a csodához?
     - Ha nem, akkor eldobja a kártyát
   - Ha igen, akkor megveszi a csodát és a kártya alá kerül (az árat itt kell kiszámolni)

Ha az akciók listája kiürül, akkor vége a játéknak. Figyeljük a katonai győzelemre is!

```
game :: Player -> Player -> [Action] -> [Table] -> Name
```

Tesztek

```
game (P "Cecil" [] [] 7) (P "Dezső" [] [] 7) [] [] == "Draw"

game (P "Cecil" [] [] 7) (P "Dezső" [] [] 7) [(DropCard (Materials "Shelf quar

game (P "András" [] [((W "The Hanging Gardens" ([(Papyrus 1), (Glass 1), (Wood

game (P "András" [] [((W "The Hanging Gardens" ([(Papyrus 1), (Glass 1), (Wood

game (P "Adrienn" [] [((W "The Hanging Gardens" ([(Papyrus 1), (Glass 1), (Woo

game (P "Adrienn" [] [((W "The Hanging Gardens" ([(Papyrus 1), (Glass 1), (Woo
```

Megoldás

Letöltés

Név:	nagybeadando.zip
Feltöltés ideje:	2024. 05. 14. 19:17
Értékelés:	5
Státusz:	Elfogadva
Feltöltések száma:	15
Értékelte:	Erdei Zsófia
Megjegyzések:	

Automatikus tesztelés eredményei

#1

```
Elért pontszám: 6/14 pont.

Teszteken sikeresen átmenő definíciók: countBasicPoints, militaryVictory, costInMoney,
guildPoints, scientificPlusPoints, canBuyWonder, buildWonder.

A következő konstansokat, függvényeket és adattípusokat nem találni a megoldásban:
isCardFree, cardToNothing, game
Megbukott tesztek:
```