Bradavice

Vaším úkolem je implementovat modul, který bude umožňovat stáhnout seznam postav z Harry Potter API a bude implementovat některé funkce pro práci s tímto seznamem. Toto API, které poskytuje seznam postav ze světa Harry Potter, najdete na adrese https://www.fi.muni.cz/~xjonas/hpcharacters.json. Jedná se o službu, která na HTTPS dotaz vrací jeden dokument ve formátu JSON.

Odevzdávat budete jeden soubor s modulem `Hogwarts`, který definuje a exportuje datový typ `DB` a následující funkce:

```
getDB :: IO (Either String DB)
getActor :: String -> DB -> Maybe String
getPatronus :: String -> DB -> Maybe String
getMugglebornNames :: DB -> [String]
getHouseCounts :: DB -> [(String, Int)]
```

Akce `getDB` se připojí k Harry Potter API a pokusí se z něj stáhnout databázi postav. Pokud stažení proběhne bez problémů, výsledek `IO` akce bude `Right db`, kde `db` je databáze postav. Pokud stažení z nějakého důvodu selže, například kvůli problému se sítí nebo kvůli nevalidnímu JSON souboru, výsledkem akce bude `Left error`, kde `error` je textový popis chyby, která nastala.

Zbývající funkce už pracují se staženou databází studentů a *jsou čisté*. Jejich účel byste pravděpodobně dokázali odvodit z jejich názvu a typu, ale pro úplnost:

- Funkce `getActor` dostane jméno postavy a databázi postav a vrátí jméno herce, který zadanou postavu hrál ve filmech. Pokud zadaná postava neexistuje nebo ve filmech nebyla, funkce vrátí `Nothing`.
- Funkce `getPatronus` dostane jméno postavy a databázi postav a vrátí textový popis *patrona* zadané postavy. Pokud zadaná postava neexistuje, nemá známého patrona nebo patrona nemá vůbec, funkce vrátí `Nothing`.
- Funkce `getMugglebornNames` dostane databázi postav a vrátí seznam jmen postav, které jsou *kouzelníci z mudlovské rodiny* (muggleborn).[1]
- Funkce `getHouseCounts` dostane databázi postav a vrátí počty postav z jednotlivých kolejí (house). Konkrétně vrátí seznam dvojic `(house, n)`, kde `house` je jméno koleje a `n` je počet postav z této koleje.

Cílem úlohy je, abyste nastudovali, nainstalovali a použili existující knihovny pro jazyk Haskell. Konkrétně se vám můžou hodit následující dva moduly:

- Modul `Network.HTTP.Simple` z balíčku `http-conduit` pro jednoduchou práci s HTTP(S) požadavky.
- Modul `Data.Aeson` z balíčku `aeson` pro práci s daty ve formátu JSON.

Příklady

Pro ověření funkčnosti vaší implementace nabízíme několik příkladů, jak by se vaše funkce měly chovat:

```
ghci> Right db <- getDB

ghci> getActor "Minerva McGonagall" db
Just "Dame Maggie Smith"
ghci> getActor "Jiri Barnat" db
Nothing
ghci> getPatronus "Luna Lovegood" db
Just "hare"
ghci> getPatronus "Draco Malfoy" db
Nothing
```

[1] Prosíme, nepoužívejte tuto funkci na špatné účely.

```
ghci> take 4 $ getMugglebornNames db
["Hermione Granger","Lily Potter","Justin Finch-Fletchley","Colin Creevey"]
ghci> getHouseCounts db
[("Gryffindor",47),("Hufflepuff",19),("Ravenclaw",22),("Slytherin",46)]
```

Pokud se odpojíte od internetu a pokusíte se použít API, měli byste dostat smysluplnou chybovou hodnotu:

```
ghci> d <- getDB
ghci> print d
Left "Unable to connect to the API."
```

Rady na závěr

Samozřejmě nejen můžete, ale měli byste, definovat vlastní pomocné funkce. Ty ale nebudou z modulu `Hogwarts` exportované, a tudíž viditelné pro uživatele.

Můžete si všimnout, že zadání nevynucuje konkrétní implementaci datového typu `DB`. Tedy můžete použít libovolný datový typ, který stačí na zodpovězení dotazů `getActor`, `getPatronus`, `getMugglebornNames` a `getHouseCounts`. Zejména vaše interní reprezentace postav nemusí obsahovat všechny informace z Harry Potter API. Pokud se však rozhodnete některé informace vynechat, je potřeba napsat vlastní instanci typové třídy `FromJSON` pro váš datový typ reprezentující postavy, protože ta automaticky odvozená nebude odpovídat formátu zadaného API.

Časovou složitost vyhledávání v databázi nemusíte optimalizovat (pokud chcete, samozřejmě můžete).

Moduly `Network.HTTP.Simple` i `Data.Aeson` nereprezentují řetězce pomocí datového typu `String`, ale pomocí datového typu `ByteString`, protože ten je paměťově efektivnější. Datový typ `ByteString` může být buď striktní (`Data.ByteString.ByteString`), nebo líný (`Data.ByteString.Lazy.ByteString`) a oba moduly `Network.HTTP.Simple` a `Data.Aeson` podporují obě tyto varianty. Dejte si tedy pozor, abyste nepomíchali striktní a líné varianty typu `ByteString`. Jednoduché řešení, které doporučujeme, je používat z obou modulů jen funkce, které pracují se striktní variantou typu `ByteString`.

Nebojte se často kontrolovat dokumentaci používaných modulů na Hackage či využívat vyhledávač Hoogle. V případě problémů a dotazů se nebojte obrátit se na nás například na diskuzním fóru.