IB015 – Domácí úkol 4: Databáze seriálů

Termín: 30. října 2019 23.59; způsob odevzdání je popsán níže.

V tomto domácím úkolu si vyzkoušíte vytvořit o něco delší kód v Haskellu. Odměnou vám za to mohou být až dva body do hodnocení.

Vaším úkolem je naprogramovat několik funkcí pracujících s databází seriálů a jejich dílů. Úkol se zaměřuje na používání seznamových funkcí, zejména `map` a `filter`, a na skládání standardních i vlastních funkcí. Většinu podúkolů lze vyřešit na tři řádky nebo méně včetně lokálních definic a žádné řešení nepotřebuje explicitní rekurzi.

Kostru domácí úlohy si můžete stáhnout ze studijních materiálů.

Reprezentace databáze

Databáze obsahuje dvě tabulky – jedna obsahuje informace o seriálech jako celcích, druhá o jednotlivých epizodách. Jejich relační schémata jsou (primární klíče italikou):

- Shows (*ID seriálu*, Jméno seriálu, Celkové hodnocení)
- Episodes (*ID seriálu*, *Pořadí epizody v seriálu*, Rok vydání, Jméno epizody)

Primární klíč říká, čím je možné jednoznačně identifikovat záznam v tabulce. V tomto případě má každý seriál unikátní identifikační číslo a každá epizoda je jednoznačně určena kombinací identifikačního čísla seriálu a jejího pořadového čísla v rámci seriálu. Různé epizody se mohou shodovat v čísle seriálu nebo v pořadovém čísle, nikdy ale ne v obojím.

Tabulky jsou reprezentovány jako seznamy n-tic. Identifikační čísla, pořadové číslo, rok vydání a hodnocení jsou typu `Int` a jména jsou řetězce (`String`). Protože v Haskellu nemůžeme mít globální měnitelný stav, je databáze (resp. jen potřebná část) vždy předávána funkcím jako argument.

Můžete předpokládat, že ve vstupních tabulkách (tj. těch předávaných jako argument) má každý záznam skutečně unikátní hodnotu primárního klíče. Navíc můžete předpokládat, že každé identifikační číslo seriálu, které se vyskytuje v tabulce epizod, je k nalezení také v tabulce seriálů. Naopak to neplatí – seriál nemusí mít v databázi žádnou epizodu (třeba je zatím jen ohlášený). Identifikační čísla seriálů jsou nezáporná. Nic jiného o databázi nepředpokládejte – rozsahy jiných hodnot, počet a pořadí záznamů apod.

Aby byla z typů lépe poznat sémantika funkcí, zavádíme v kostře několik typových aliasů. Například řádek `type ShowID = Int` znamená, že kdykoli se v typové signatuře objeví `ShowID`, je to totéž, jako by místo toho byl `Int`. Něco podobného už znáte – `String` je typovým aliasem pro `[Char]`. Aliasy z kostry se objevují i v typových signaturách dále v tomto zadání a jsou to tyto:

```
type ShowID = Int
type ShowName = String
type Rating = Int
type EpisodeNumber = Int
type Year = Int
type EpisodeName = String
type ShowInfo = (ShowID, ShowName, Rating)
type EpisodeInfo = (ShowID, EpisodeNumber, Year, EpisodeName)
type Shows = [ShowInfo]
type Episodes = [EpisodeInfo]
```

Aby se vám s n-ticemi lépe pracovalo a výsledný kód byl čitelnější, vřele doporučujeme napsat si funkce vracející jednotlivé složky (tzv. *gettery*); například `episodeGetShowId :: EpisodeInfo -> ShowID`.

Funkce k implementaci

Následuje popis funkcí, které máte implementovat. V příkladech se držíme zavedeného formátu, v němž řádky začínající znakem „>“ označují vstup interpretu a ostatní řádky jeho výstup. Tabulky použité v příkladech najdete v kostře a na konci zadání. Pro lepší čitelnost jsme zde výstupy rozdělili na řádky, nelekněte se proto, že vám interpret vypíše vše na jeden dlouhý řádek.

- **`findShowId :: Shows -> ShowName -> ShowID`**

  V databázi nalezne identifikační číslo seriálu se zadaným jménem. Pakliže se seriál daného jména v databázi nenachází, je výsledkem číslo -1. Nachází-li se seriálů s daným jménem naopak víc, vrací identifikační číslo prvního z nich.

```
> findShowId shs "Mentalista"
45
> findShowId shs "Rišo a Morty"
-1
```

- **`getEpisodes :: Episodes -> ShowID -> Episodes`**

  Z databáze vybere všechny epizody seriálu se zadaným identifikačním číslem. Pořadí epizod ve výsledném seznamu je stejné jako ve vstupním seznamu.

```
> getEpisodes eps 48
[(48,8,2010,"Smutecni rec"), (48,4,2009,"Dovolena")]
> getEpisodes eps 66
[]
```

- **`countEpisodes :: Episodes -> ShowID -> Int`**

  Vrátí počet epizod seriálu se zadaným identifikačním číslem.

```
> countEpisodes eps 51
3
> countEpisodes eps 66
0
```

- **`isShowContiguous :: Episodes -> ShowID -> Bool`**

  Zjistí, zda epizody seriálu se zadaným identifikačním číslem, které se nachází v databázi, tvoří souvislou řadu. To znamená, že v databázi nechybí žádné epizody mezi nejstarší a nejnovější. Kritériem je číslo epizody v rámci seriálu. Seriály bez epizod jsou považovány za souvislé.

  Povšimněte si, že funkce nezjišťuje, zda je seriál v databázi *kompletní*. Případné chybějící epizody na začátku a na konci nemají na souvislost vliv. Nezapomeňte, že o pořadí epizod v databázi nemůžete nic předpokládat.

```
> isShowContiguous eps 4
False
> isShowContiguous eps 51
True
> isShowContiguous eps 129
True
```

- **`publicationRange :: Episodes -> ShowID -> (Year, Year)`**

  Vrátí interval, v němž se seriál s daným identifikačním číslem vysílal. Složkami dvojice jsou postupně rok vydání nejstarší a rok vydání nejmladší epizody zadaného seriálu.

  Můžete předpokládat, že se v databázi vyskytuje alespoň jedna epizoda zadaného seriálu.

```
> publicationRange eps 4
(2006,2006)
> publicationRange eps 45
(2008,2013)
```

- **`bestRating :: Shows -> ShowName`**

  Vrátí jméno nejlépe hodnoceného seriálu. Můžete předpokládat neprázdnou databázi.

  Pokud se o první místo dělí více seriálů, vraťte jméno libovolného z nich. O rozsahu hodnocení nic nepředpokládejte.

```
> bestRating shs
"Ajtaci"
```

- **`worstRating :: Shows -> ShowName`**

  Vrátí jméno nejhůře hodnoceného seriálu. Můžete předpokládat neprázdnou databázi.

  Snažte se zamezit duplikaci kódu z předchozí funkce. Pokud se o poslední místo dělí více seriálů, vraťte jméno libovolného z nich.

```
> worstRating shs
"Mentalista"
```

- **`sortByYearOfPublication :: Episodes -> Episodes`**

  Seřadí databázi epizod podle roku vydání od nejstarších po nejnovější. Epizody se stejným rokem vydání přitom zůstanou **ve stejném vzájemném pořadí**; této vlastnosti se říká *stabilní řazení*.

  Tip: Rozhodně nemusíte psát vlastní řadicí funkci. Porozhlédněte se v dokumentaci k seznamovým funkcím. Ujistěte se ale, že vámi zvolená funkce řadí *stabilně*.

```
> sortByYearOfPublication eps
[(4,1,2006,"Vcerejsi odpad")
,(4,4,2006,"Cervene dvere")
,(4,6,2006,"Prijizdi teta Irma")
,(4,2,2006,"Kalamity Jen")
,(45,2,2008,"Rude vlasy a stribrna paska")
,(48,4,2009,"Dovolena")
,(45,56,2010,"Vesely rudy skritek")
,(48,8,2010,"Smutecni rec")
,(45,77,2011,"Rude probleskujici svetlo")
,(45,124,2013,"Rudy John")
,(51,4,2019,"")
,(51,3,2019,"")
,(51,5,2019,"")]
```

- **`showEpisodes :: Shows -> Episodes -> [(ShowName, [EpisodeName])]`**

  Vrátí seznam jmen seriálů a ke každému i seznam jmen epizod onoho seriálu. Pořadí seriálů ve výsledku přesně odpovídá jejich pořadí ve vstupním seznamu. Stejně tak vzájemné pořadí epizod v rámci jednoho seriálu odpovídá jejich pořadí ve vstupním seznamu. Seriály bez epizod se ve výsledku objeví s prázdným seznamem.

```
> showEpisodes shs eps
[("Ajtaci",["Vcerejsi odpad",
            "Cervene dvere",
            "Prijizdi teta Irma",
            "Kalamity Jen"])
,("Mentalista",["Vesely rudy skritek",
                "Rudy John",
                "Rude vlasy a stribrna paska",
                "Rude probleskujici svetlo"])
,("Miranda",["Smutecni rec",
             "Dovolena"])
,("Most!",["", "", ""])
,("Zdivocela zeme",[])]
```

- **`join :: Shows -> Episodes -> [(ShowName, EpisodeName)]`**

  Ke každé epizodě vrátí její jméno (jako **druhou** složku) a jméno seriálu, z něhož pochází (jako **první** složku). Na pořadí dvojic ve výsledném seznamu tentokrát nezáleží. Seriály bez epizod se ve výsledku neobjeví.

V relační algebře bychom tuto operaci zapsali jako $\Pi_{(\textit{Jméno seriálu},\ \textit{Jméno epizody})}(\textit{Shows} \bowtie \textit{Episodes})$. Z použitého přirozeného spojení vychází i jméno funkce.

```
> join shs eps
[("Ajtaci","Vcerejsi odpad")
,("Ajtaci","Cervene dvere")
,("Ajtaci","Prijizdi teta Irma")
,("Ajtaci","Kalamity Jen")
,("Mentalista","Vesely rudy skritek")
,("Mentalista","Rudy John")
,("Mentalista","Rude vlasy a stribrna paska")
,("Mentalista","Rude probleskujici svetlo")
,("Miranda","Smutecni rec")
,("Miranda","Dovolena")
,("Most!","")
,("Most!","")
,("Most!","")]
```

Poznámky a tipy

- Směle využívejte funkcí z `Prelude` a `Data.List`.
- Importovat smíte i jiné moduly z balíku `base`, pohodlně se bez nich ale obejdete.
- Neduplikujte kód! Snažte se vždy využít funkce, které jste již naprogramovali. Pokud to nejde přímo, ale přesto vidíte v řešení podobu, vytkněte podobnou část do pomocné funkce.
- Zkuste se zamyslet, jestli se nedá využít toho, že funkce `(<=)` porovnává n-tice *lexikograficky*. Jinými slovy, n-tice se primárně porovnávají podle první složky; pokud mají první složku stejnou, pak podle druhé složky; pokud mají i druhou složku stejnou, pak podle třetí složky, a tak dále. Stejný způsob se pak vztahuje i na knihovní funkce, které jsou implementované pomocí funkce `(<=)`, tedy například funkce jako `max`, `maximum` a podobně.
- U všech funkcí uvádějte jejich typové signatury.
- Funkce jsou v kostře zadefinovány jako `undefined`, takže projdou překladem, ale jejich zavolání způsobí chybu. Neimplementujete-li některou funkci, neodstraňujte ji, ale nechte ji rovnu `undefined`.
- Nejste-li si jisti nějakou částí zadání, zeptejte se v diskusním fóru.
- Přebíráte-li kód odjinud, uveďte zdroj, jinak bude na vaši práci pohlíženo jako na plagiát.
- Nezapomeňte, že **opisování je zakázáno** a bude postihováno podle disciplinárního řádu.
- Než řešení odevzdáte, **pečlivě si přečtěte následující sekci** a ujistěte se, že váš kód splňuje všechny náležitosti. Neztrácejte body jen kvůli nepozornému čtení pokynů.

Odevzdání

Tento domácí úkol se neodevzdává přes odpovědník, nýbrž přes odevzdávárny v Informačním systému. O tom, kterou odevzdávárnu máte použít, rozhoduje číslo vaší seminární skupiny. Do odevzdávárny vkládejte **jediný** soubor s příponou `.hs` obsahující vaši implementaci **všech** požadovaných funkcí. Nebudou-li v souboru přítomny všechny funkce, testy selžou a vy nedostanete žádné body. Totéž vás čeká, pokud odevzdaný soubor nebude lze přeložit překladačem GHC 8.6. Doporučujeme vám si jej proto před odevzdáním zkusit spustit na Aise (nezapomeňte přidat modul s novým GHC). **Všechny funkce musí mít typovou signaturu**. V odevzdaném souboru neuvádějte hlavičku `module` (pokud nevíte, o co se jedná, vůbec to nevadí).

Vyhodnocení po nahrání souboru **není** okamžité; automatický testovací nástroj kontroluje soubory v odevzdávárně několikrát denně. Podle času odevzdání a vytížení vyhodnocovacího serveru může vyhodnocení trvat několik desítek minut. Po vyhodnocení se získané body a případný výpis neprošedších testů objeví v poznámkovém bloku. U nesprávně implementovaných funkcí se dozvíte příklad vstupu, na němž se váš výsledek neshoduje s očekávaným.

Máte **pět možností odevzdání**, započítává se nejlepší z nich. Další odevzdání provedete tak, že do odevzdávárny nahrajete novou verzi, jíž přepíšete odevzdaný soubor. Vzhledem k prodlevám při vyhodnocování neodkládejte práci na poslední chvíli, ať možnost vícenásobného odevzdání v případě potřeby vůbec stihnete využít. S blížícím se termínem uzavření odevzdáváren očekávejte větší (i několikahodinové) prodlevy.

S odevzdávárnou zacházejte s rozvahou, abyste nepřišli o možnosti odevzdání. I když nahrajete nové řešení ještě před zveřejněním výsledku v poznámkovém bloku, vyhodnocovací nástroj už může mít vaše dřívější odevzdání ve frontě. Z jeho pohledu tak došlo ke dvěma odevzdáním a vy si vyplýtváte jeden pokus. Podobně se vám mohou započítat odevzdání navíc, pokud do odevzdávárny omylem vložíte více než jeden soubor.

Hodnocení

Za funkčnost můžete od automatických testů obdržet **až 1,8 bodu** podle toho, které funkce se vám podařilo správně implementovat. Za částečně implementované funkce (např. nefunguje některý okrajový případ) žádné body nezískáte.

Řešení budou po termínu odevzdávání hodnotit cvičící. Ti vám poskytnou zejména zpětnou vazbu na kód, ale také vám za úhlednost a pochopitelnost řešení mohou udělit další dvě desetiny bodu. Neočividné či zajímavé části řešení proto stručně komentujte v kódu. Tipy k psaní hezkého kódu naleznete v diskusním fóru.

V součtu tedy můžete za úlohu získat **až 2 body**.

Testovací data

Následující tabulky jsou použity v příkladech vyhodnocení funkcí výše a můžete je použít k vlastnímu testování. Automatické testy ale probíhají nad jinými (automaticky generovanými) daty, řešení šité na míru těm zdejším vám proto žádné body nevynese.

```
shs :: Shows
shs = [(  4, "Ajtaci",          88)
      ,( 45, "Mentalista",      70)
      ,( 48, "Miranda",         84)
      ,( 51, "Most!",           84)
      ,(129, "Zdivocela zeme", 71)
      ]

eps :: Episodes
eps = [( 4,   1, 2006, "Vcerejsi odpad")
      ,(45,  56, 2010, "Vesely rudy skritek")
      ,( 4,   4, 2006, "Cervene dvere")
      ,(45, 124, 2013, "Rudy John")
      ,(48,   8, 2010, "Smutecni rec")
      ,(51,   4, 2019, "")
      ,( 4,   6, 2006, "Prijizdi teta Irma")
      ,( 4,   2, 2006, "Kalamity Jen")
      ,(51,   3, 2019, "")
      ,(45,   2, 2008, "Rude vlasy a stribrna paska")
      ,(48,   4, 2009, "Dovolena")
      ,(45,  77, 2011, "Rude probleskujici svetlo")
      ,(51,   5, 2019, "")
      ]
```