IB015 – Domácí úkol 4: Databáze onemocnění

Termín odevzdání: 29. 11. 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 3 body do hodnocení.

Vaším úkolem je naprogramovat několik funkcí pracujících s databází osob, onemocnění a jejich ukončení. 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 tři tabulky – jedna obsahuje informace o osobách, druhá popisuje onemocnění (tedy kombinaci nemoci a osoby, která danou nemocí trpěla nebo stále trpí), a poslední tabulka zase popisuje východisko nemoci (způsob ukončení). Východiskem nemoci je buď vyléčení nemocné osoby, nebo její smrt. Pokud není onemocnění ukončené, tak je osoba stále nemocná.

Jeden řádek každé tabulky je uložen jako jedna hodnota níže definovaného datového typu. Relační schémata těchto tabulek jsou následující (primární klíče, jednoznačné identifikátory záznamu v tabulce, jsou *znázorněny italikou*):

- Persons: (*ID osoby*, Jméno osoby)
- Infections: (*ID onemocnění*, ID osoby, Jméno onemocnění)
- Terminations: (*ID onemocnění*, Typ ukončení onemocnění)

V tomto případě má každá osoba a každé onemocnění unikátní identifikační číslo. Tzn. o každé ze vstupních tabulek (tj. těch předávaných jako argument) můžete předpokládat, že má každý záznam skutečně unikátní hodnotu primárního klíče.

Jedna osoba může mít libovolně mnoho nemocí (i žádnou), pro každou pak bude v tabulce samostatný záznam. Pro každé onemocnění bude, v tabulce o ukončení onemocnění, buď jeden, nebo žádný záznam o ukončení.

Můžete předpokládat, že každé identifikační číslo osoby, které se vyskytuje v tabulce onemocnění, je k nalezení také v tabulce osob. Naopak to neplatí – osoba nemusí mít v databázi žádné onemocnění (osoba zatím nikdy neonemocněla).

Posledním předpokladem je, že každé identifikační číslo onemocnění v tabulce ukončení onemocnění se vyskytuje v tabulce onemocnění.

Nic jiného o databázi nepředpokládejte – rozsahy hodnot, počet a pořadí záznamů a podobně nejsou specifikovány.

Tabulky jsou reprezentovány jako seznamy položek, typ těchto položek je definován níže. Identifikační čísla jsou typu `Integer` a jména jsou řetězce (`String`). Protože v Haskellu nemůžeme mít globální měnitelný stav, je databáze (resp. jen potřebná část) vždy předávána funkcím jako argument.

Aby byl z typů lépe poznat význam funkcí, zavádíme v kostře několik typových aliasů. Aliasy z kostry se objevují i v typových signaturách dále v tomto zadání a jsou to:

```
type PersonId = Integer
type PersonName = String
type InfectionId = Integer
type InfectionName = String
```

Dále si zadefinujeme i nějaké vlastní datové typy. Prvně si definujeme typ `TerminationType`, který určuje, jak bylo onemocnění ukončeno:

```
data TerminationType = Recovered | Dead
```

Dále si zadefinujeme typy pro řádky jednotlivých tabulek a tabulky samotné:

```haskell
data Person = Person PersonId PersonName
data Infection = Infection InfectionId PersonId InfectionName
data Termination = Termination InfectionId TerminationType

type Persons = [Person]
type Infections = [Infection]
type Terminations = [Termination]
```

Aby se vám se záznamy lépe pracovalo a výsledný kód byl čitelnější, vřele doporučujeme napsat si funkce vracející jednotlivé složky (tzv. *gettery*); například `personId :: Person -> PersonId`.

Funkce k implementaci

Následuje popis funkcí, které máte implementovat. 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.

- **`countOfInfected :: Infections -> InfectionName -> Int`**

  Vrátí počet osob, které někdy prodělaly danou nemoc. Nezáleží na tom, jestli jsou ještě pořád nemocné.

  ```haskell
  countOfInfected infs "COVID" ~>* 3
  countOfInfected infs "mrtvice" ~>* 0
  countOfInfected [Infection 211 16 "COVID", Infection 119 16 "COVID"] "COVID" ~>* 1
  ```

- **`dead :: Terminations -> [InfectionId]`**

  Z databáze vybere všechna ID onemocnění, která byla ukončena smrtí pacienta. Pořadí ID onemocnění ve výsledném seznamu je stejné jako ve vstupní databázi.

  ```haskell
  dead ters ~>* [2020, 128]
  ```

- **`activeCases :: Infections -> Terminations -> Infections`**

  Vrátí všechna onemocnění, která ještě nebyla ukončena (ani smrtí, ani vyléčením). Pořadí onemocnění ve výsledném seznamu musí být stejné, jako bylo ve vstupní databázi.

  Doplnění: V této úloze pro zjednodušení neuvažujte živost pacienta, pouze to, zda existuje ukončení pro danou infekci. Vracejte tedy i infekce mrtvých pacientů.

  Nápověda: Vzorové řešení používá pomocnou lokální definici `inactive :: [InfectionId]`.

  ```haskell
  activeCases infs ters
      ~>* [Infection 1 5 "COVID"
          ,Infection 5 128 "rymicka"
          ,Infection 3 5 "astma"
          ]
  activeCases [Infection 1 42 "mrtvice", Infection 2 42 "rymicka"] [Termination 1 Dead]
      ~>* [Infection 2 42 "rymicka"]
  ```

- **`somebodyDied :: Infections -> Terminations -> [InfectionName]`**

  Výsledkem funkce je seznam jmen onemocnění, na které zemřela alespoň jedna osoba. Na pořadí jmen onemocnění ve výsledném seznamu nezáleží. Zároveň se ve výsledném seznamu nevyskytují duplicitní názvy.

  ```haskell
  somebodyDied infs ters ~>* ["COVID", "zapal plic"]
  somebodyDied infs [Termination 2019 Dead, Termination 2020 Dead]
      ~>* ["COVID"]
  ```

- `checkDeaths :: Infections -> Terminations -> Bool`

  Zkontroluje, jestli každá mrtvá osoba zemřela právě na jednu nemoc.

```
checkDeaths infs ters ~>* True
checkDeaths infs [Termination 3 Dead, Termination 128 Dead] ~>* False
```

- `diseases :: Persons -> Infections -> [(PersonName, [InfectionName])]`

  Výsledkem této funkce je seznam dvojic, kde pro každou osobu v databázi osob je v druhé složce dvojice seznam jmen onemocnění, kterými tato osoba trpí nebo trpěla. V případě, že osoba nemá nebo neměla žádnou nemoc, zůstane seznam v druhé složce dvojice prázdný.

  Poznámka: Nezáleží, jestli dané onemocnění je nebo není ukončeno.

  Pořadí dvojic musí být shodné s pořadím osob ve vstupní databázi osob. Pořadí jmen onemocnění je shodné s pořadím, které je ve vstupní tabulce onemocnění. V seznamu onemocnění dané osoby mohou být duplicity (daná osoba mohla po vyléčení nemoci onemocnět znovu na tutéž nemoc).

```
diseases pers infs
    ~>* [("Augustin", ["COVID", "astma"])
        ,("Baltazar", [])
        ,("Ctirad", ["COVID"])
        ,("Zdenek", ["rymicka"])
        ,("Drahoslav", ["COVID", "astma", "zapal plic"])
        ]
diseases pers [Infection 42 1 "rymicka", Infection 21 1 "rymicka"]
    ~>* [("Augustin", ["rymicka", "rymicka"])
        ,("Baltazar", [])
        ,("Ctirad", [])
        ,("Zdenek", [])
        ,("Drahoslav", [])
        ]
```

Poznámky a tipy

- Směle využívejte funkcí z `Prelude` a `Data.List`.
- Importovat smíte i jiné moduly z balíku `base`,[1] 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.
- Funkce jsou v kostře zakomentovány a zadefinovány jako `undefined`, takže po odkomentování projdou překladem, ale jejich zavolání způsobí chybu. Pokud nějakou funkci neimplementujete, můžete ji nechat zakomentovanou.
- Nejste-li si jisti nějakou částí zadání, zeptejte se v diskusním fóru.
- Nezapomeňte, že **opisování je zakázáno** a bude postihováno podle disciplinárního řádu.
- Přebíráte-li kód odjinud, uveďte zdroj, jinak bude na vaši práci pohlíženo jako na plagiát.
- 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 příslušnou odevzdávárnu 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.
  - Pokud chcete odevzdat znovu, můžete soubor přepsat či přidat nový, nezáleží na tom – vyhodnocuje se vždy nejnovější odevzdaný soubor.
  - Žádné jiné soubory nevkládejte.
  - Již vložené soubory nepřejmenovávejte, mohou se pak vyhodnotit dvakrát.

[1] Výjimku tvoří moduly které jsou „Unsafe“, ty však určitě nebudete potřebovat (v hlavičce dokumentace jsou označeny jako „Safe Haskell: Unsafe“).

- Vaše řešení **musí zachovat všechny definice datových typů tak, jak jsou v zadání**, jediná povolená modifikace je přidání instance typové třídy.
- Řešení musí jít přeložit překladačem GHC 8.10. Zkuste si jej proto před odevzdáním 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).

Vyhodnocování

- Vyhodnocení po nahrání souboru **není** okamžité.
  - Automatický testovací nástroj kontroluje soubory v odevzdávárně v pravidelných intervalech několika minut. Podle času odevzdání a vytížení vyhodnocovacího serveru může vyhodnocení trvat několik desítek minut.
  - Pokud se vám výsledky neobjevily cca do půl hodiny a neblíží se zatím čas deadline, dejte nám vědět ve fóru.
- 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.
  - Nejprve jsou v bloku uvedeny vstupy, následně výstup učitelského řešení a váš výstup.
  - V bloku uvidíte dvě sady testů – *build* a *test*. Může se stát, že se zobrazí nejprve jen *build*, do pěti minut by pak měly přibýt výsledky samotných testů funkčnosti.
- Máte **pět možností odevzdání**, započítává se nejlepší z nich.
  - Pokud v odevzdání neprojde kontrola syntaxe, tak se do tohoto limitu nepočítá, pokud však kontrola projde, je automaticky započítáno a nelze to zvrátit.
- Další odevzdání provedete tak, že do odevzdávárny nahrajete novou verzi.
- 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.
  - Není žádná garance rychlosti vyhodnocování (může se tedy například stát, že výsledky řešení odevzdaného v neděli ve 21.00 neuvidíte do konce deadline).

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 (a pravděpodobně má) 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 nebo pokud soubor přejmenujete (každý soubor se vezme jako samostatné odevzdání).

Hodnocení

Za funkčnost můžete od automatických testů obdržet **až 2,5 bodu** podle toho, které funkce se vám podařilo správně implementovat. Za částečně implementované funkce (např. nefunguje některý okrajový případ) žádné body nezískáte.

Následně vám cvičící dají zpětnou vazbu na kód, ale také vám za úhlednost a pochopitelnost řešení mohou udělit dalšího 0,5 bodu. Snažte se proto kód psát hezký a případné neočividné či zajímavé části řešení stručně komentujte v kódu. Tipy k psaní hezkého kódu naleznete ve sbírce. Hodnotit na kvalitu se bude poslední odevzdání s maximem bodů ze všech vašich odevzdání. Pokud vám tedy po dosažení plného počtu automaticky přidělovaných bodů ještě zbývají pokusy, můžete bezpečně zkusit svůj kód zkrášlit.

V součtu tedy můžete za úlohu získat **až 3 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. Doporučujeme tedy funkce poctivě otestovat.

```haskell
pers :: Persons
pers = [Person 1 "Augustin"
       ,Person 2 "Baltazar"
       ,Person 42 "Ctirad"
       ,Person 128 "Zdenek"
       ,Person 5 "Drahoslav"
       ]

infs :: Infections
infs = [Infection 2020 1 "COVID"
       ,Infection 2019 42 "COVID"
       ,Infection 1 5 "COVID"
       ,Infection 5 128 "rymicka"
       ,Infection 3 5 "astma"
       ,Infection 2 1 "astma"
       ,Infection 128 5 "zapal plic"
       ]

ters :: Terminations
ters = [Termination 2020 Dead
       ,Termination 2 Recovered
       ,Termination 2019 Recovered
       ,Termination 128 Dead
       ]
```
