Домашна работа № 4 по Функционално програмиране

специалност „Компютърни науки“, II курс, I поток, 2021/2022 учебна година

Решенията трябва да са готови за автоматично тестване. Важно е програмният код да бъде добре форматиран и да съдържа коментари на ключовите места. Предайте решенията на всички задачи в *един* файл с наименование *hw4_<FN>.hs*, където *<FN>* е Вашият факултетен номер.

Домашните работи се предават като изпълнение на съответното задание в курса по ФП в Moodle (https://learn.fmi.uni-sofia.bg/course/view.php?id=7484) най-късно до 23:55 ч. на 20.01.2022 г. (четвъртък).

Приятна работа и успех!

Задача 1. Нека е дефинирана следната структура на база от данни:

```
type Name = String
type Date = String
type Class = String
type Result = String
type Launched = Int

data Battle = Battle Name Date deriving Show
data Ship = Ship Name Class Launched deriving Show
data Outcome = Outcome Name Name Result deriving Show

type Database = ([Outcome], [Battle], [Ship])
```

Алгебричният тип **`Battle`** представя данните за дадена битка - нейното име и датата, на която се е състояла. Алгебричният тип **`Ship`** представя данните за даден кораб - неговото име и годината на пускането му във вода. Алгебричният тип **`Outcome`** представя данните за резултата за даден кораб от дадена битка - името на кораба, името на битката и резултата за кораба - дали е бил повреден (damaged), потънал (sunk), или неповреден (ok).

Да се дефинират следните функции:

- функция **`getSunk :: Database -> [(Name, [Name])]`**, която получава като аргумент база от данни и връща имената на всички кораби, потънали в битка, под формата на списък от двойки от вида (<име на битка>,<списък от имена на кораби, потънали в тази битка>);

- функция **`inBattleAfterDamaged :: Database -> [Name]`**, която получава като аргумент база от данни и връща списък от имената на тези кораби, които са били повредени в една битка, но по-късно са участвали в друга битка.

Примери:

```
outcomes :: [Outcome]
outcomes = [ Outcome "Bismarck" "North Atlantic" "sunk",
Outcome "California" "Surigao Strait" "ok", Outcome "Duke of
York" "North Cape" "ok", Outcome "Fuso" "Surigao Strait"
"sunk", Outcome "Hood" "North Atlantic" "sunk", Outcome "King
George V" "North Atlantic" "ok", Outcome "Kirishima"
"Guadalcanal" "sunk", Outcome "Prince of Wales" "North
Atlantic" "damaged", Outcome "Rodney" "North Atlantic" "ok",
Outcome "Schamhorst" "North Cape" "sunk", Outcome "South
Dakota" "Guadalcanal" "damaged", Outcome "Tennessee" "Surigao
Strait" "ok", Outcome "Washington" "Guadalcanal" "ok", Outcome
"Prince of Wales" "Guadalcanal" "ok", Outcome "West Virginia"
"Surigao Strait" "ok", Outcome "Yamashiro" "Surigao Strait"
"sunk", Outcome "California" "Guadalcanal" "damaged" ]

battles :: [Battle]
battles = [ Battle "Guadalcanal" "1942-11-15", Battle "North
Atlantic" "1941-05-25", Battle "North Cape" "1943-12-26",
Battle "Surigao Strait" "1944-10-25" ]

ships :: [Ship]
ships = [ Ship "California" "Tennessee" 1921, Ship "Haruna"
"Kongo" 1916, Ship "Hiei" "Kongo" 1914, Ship "Iowa" "Iowa"
1943, Ship "Kirishima" "Kongo" 1915, Ship "Kongo" "Kongo"
1913, Ship "Missouri" "Iowa" 1944, Ship "Musashi" "Yamato"
1942, Ship "New Jersey" "Iowa" 1943, Ship "North Carolina"
"North Carolina" 1941, Ship "Ramillies" "Revenge" 1917, Ship
"Renown" "Renown" 1916, Ship "Repulse" "Renown" 1916, Ship
"Resolution" "Renown" 1916, Ship "Revenge" "Revenge" 1916,
Ship "Royal Oak" "Revenge" 1916, Ship "Royal Sovereign"
"Revenge" 1916, Ship "Tennessee" "Tennessee" 1920, Ship
"Washington" "North Carolina" 1941, Ship "Wisconsin" "Iowa"
1944, Ship "Yamato" "Yamato" 1941, Ship "Yamashiro" "Yamato"
1947, Ship "South Dakota" "North Carolina" 1941, Ship
"Bismarck" "North Carolina" 1911, Ship "Duke of York" "Renown"
1916, Ship "Fuso" "Iowa" 1940, Ship "Hood" "Iowa" 1942, Ship
"Rodney" "Yamato" 1915, Ship "Yanashiro" "Yamato" 1918, Ship
```

```
"Schamhorst" "North Carolina" 1917, Ship "Prince of Wales"
"North Carolina" 1937, Ship "King George V" "Iowa" 1942, Ship
"West Virginia" "Iowa" 1942 ]

database :: Database
database = (outcomes, battles, ships)

getSunk database → [("Guadalcanal",["Kirishima"]),("North
Atlantic",["Bismarck","Hood"]),("North
Cape",["Schamhorst"]),("Surigao Strait",["Fuso","Yamashiro"])]

inBattleAfterDamaged database → ["California","Prince of
Wales"]
```

Задача 2. Нека за представянето на двоично дърво от цели числа се използва алгебричен тип със следната дефиниция:

```
data BTree = Nil | Node Int BTree Btree
```

Да се дефинира функция **`grandchildrenIncreased :: BTree -> Bool`**, която проверява дали всеки връх на двоичното дърво tree е по-голям поне с 1 от своя дядо (ако има такъв).

Примери:

```
grandchildrenIncreased t1 → True
grandchildrenIncreased t2 → False
```