THU Haskell 2019 Homework 06

截止时间：2019 年 4 月 20 日（周六）23:59:00

作业格式要求

这次作业的格式比较特殊。对 QuickCheck 一题，我们会提供一个 stack 项目，你需要编辑 `test/Spec.hs`，在里面编写若干条属性 (property) 并在 `Spec.hs` 的 `main` 函数中调用。非编程题需要将解答放到 `doc/hw6.pdf`。

```
.
├── doc
|   ├── ...
|   └── hw6.pdf
└── test
    └── Spec.hs
```

提交的项目应当能在根目录正常执行 `stack test` (能得到 pass 或 fail)。打包前请执行 `stack clean --full` 清理项目。检查作业格式的网站为 https://thufp.sonion.xyz/haskellformat/。

编程题：用 QuickCheck 做基于属性的测试

以前的编程题，是你写 `src/` 里的实现，然后我们用 `test/Spec.hs` 去测试。这次你来写测试，编写 QuickCheck 风格的属性，来尽可能地发现我们提供的*被测程序*的 bug。

被测的程序是一个*优先队列*（priority queue）的实现。优先队列是支持这些操作的数据结构：可以创建空优先队列（`empty`）；可以插入（`insert`）若干个元素到优先队列 q；可以从 q 中取出 q 包含的元素中最小的元素（`findMin`）；可以从 q 中删除其包含的最小元素（`deleteMin`）；可以把两个优先队列合并（`meld`）成一个优先队列。用*堆*（heap）实现的优先队列可以高效地 insert、findMin、deleteMin 和 meld。

我们会提供一个 stack 项目。其中最重要的可能是这个文档：`doc/PriorityQueue.html` 描述了优先队列的各个操作的应有的性质。`src/BHeap.hs` 含有一个优先队列的错误实现 `BHeap`（错得离谱）。你需要根据优先队列的各种操作的应有的性质，在 `test/Spec.hs` 编写 QuickCheck 风格的属性，用于测试 `BHeap`，揪出它的 bug。例如，这个性质应该被满足：

- 对任意整数 n，把 n 插入空的优先队列 q，然后从 q 中取出最小元素，得到的应当是 n。

据此可以写出 QuickCheck 风格的属性：

```haskell
prop_findMin_the_only_element :: Integer -> Bool
prop_findMin_the_only_element n = findMin s == n where
    s = insert n (empty :: BHeap Integer)
```

然后使用 `quickCheck prop_findMin_the_only_element` 或 `verboseCheck prop_findMin_the_only_element` 来检查这个属性在 100 个随机生成的 `Integer` 输入下是否成立。

实际上这个属性太弱，测不出 `BHeap` 的 bug。你需要按 `Spec.hs` 里的示例，写更多的、non-trivial 的属性。这些属性不仅应该测出示例里这一种 `BHeap` 实现的 bug，还应该有可能测出其他的 `BHeap` 的错误实现的 bug。注意你写的属性应该在 `Spec.hs` 的 `main` 里被使用。

评测时，我们会将你提交的项目里的 `src/BHeap.hs` 先后替换为 1 个正确的实现和 5 个错误的实现 (文件名、模块名和类型名不会变)，然后分别用你提交的 `test/Spec.hs` 来测试这些实现。如果你的测试在正确实现上全部 pass，则得到本题 50% 的分数； `test/Spec.hs` 每在一个错误实现上 fail，则多得到 10% 的分数。

由于 QuickCheck 具有随机性，可能你写的属性本来有可能测出某个 bug，但恰好没有触发 fail。对这种情况，我们会在周日（4月21日）初次评测后公布 5 个错误实现，你可以保留 QuickCheck fail 时报出的反例联系助教申诉，拿回应有的分数。

非编程题：用归纳法证明程序正确性

1. 证明对任意有限列表 `xs :: [a]` 和有限列表 `ys :: [a]`，有 `map f (xs ++ ys)` = `(map f xs) ++ (map f ys)` 成立。

   你可能需要写出 `map` 和 `(++)` 的定义并编号，以便书写每一步推导的依据。

2. 若用如下方法定义 `unzip :: [(a, b)] -> ([a], [b])`

```haskell
unzip [] = ([], [])                         (unzip.1)
unzip ((x, y) : ps) = (x : xs, y : ys)      (unzip.2)
    where (xs, ys) = unzip ps
```

(1) 证明对于所有有限列表 `ps`，有 `zip (fst (unzip ps)) (snd (unzip ps))` = `ps` 成立。

(2) 在什么条件（有关 `xs` 和 `ys`）下，`unzip (zip xs ys)` = `(xs, ys)` 成立？请给出证明。