Haskell — Queue (1) P80618_en

We want to represent queues to improve the efficiency of its *push* and *pop* operations. In order to do so, we implement queues through two lists, so that if we concatenate the first with the reverse of the second, we get the elements of the queue if exit order. Using a *Queue* constructor for this type,

$q = Queue$ [2,8,5] [4,7]

would represent que queue where the first element is 2, followed by 8, 5, 7 and 4.

In this way, the push operation is made by prepending an element to the second list (which is less expensive than appending it to its end).

On the other hand, the pop operation is now made by extracting the first element in the first queue, provided it exists. If it does not exist, all the elements in the second list are transferred to the first list (by reversing its order).

1. Implement generic queues using the following interface:

   data *Queue a = Queue* [*a*] [*a*]
      deriving (**Show**)

   create :: *Queue a*
   push :: *a* → *Queue a* → *Queue a*
   pop :: *Queue a* → *Queue a*
   top :: *Queue a* → *a*
   empty :: *Queue a* → **Bool**

2. Define equality for queues so that two queues are equal if and only if they have the same elements, in the same order, independently of their representation. In order to do so, write that queues are an instance of the **Eq** class, where (==) is the equality operation you have to define:

   instance Eq *a* ⇒ **Eq** (*Queue a*)
      where
         ...

   Observe that, in order to have *Queue a* be an instance of **Eq**, it is necessary to have that the elements of type *a* are them-selves also instances of **Eq**.

Scoring

Each section scores 50 points.

Sample input 1

```
let c = push 3 (push 2 (push 1 create))
c
top c
pop c
empty $ pop c
empty $ pop $ pop $ c
empty $ pop $ pop $ pop c
```

Sample output 1

```
Queue [] [3,2,1]
1
Queue [2,3] []
False
False
True
```

Sample input 2

```
let c1 = push 4 (pop (push 3 (push 2 (push 1 create))))
let c2 = push 4 (push 3 (push 2 create))
c1
c2
c1 == c2
```

Sample output 2

```
Queue [2,3] [4]
Queue [] [4,3,2]
True
```

Problem information

Author : Jordi Petit
Translator : Jordi Petit
Generation : 2024-05-03 00:00:27