YOUR NAME: _____

Programming in Haskell Quiz 1

Friday September 20, 2024

20 points

Rules: You are allowed to (1) use class notes and the textbook, (2) run examples in the interpreter and (3) discuss with classmates. You are NOT allowed to search the internet (most quiz functions and structures are available in hackage.)

Once you're done, type the answers into a text file and submit in Submitty.

Construct a structure `DLists` (difference lists) whose main point is to support $O(1)$ append operations.

The `DList` type definition:

```
type DList a = [a] -> [a]
```

A regular list looks like this:

```
(1:(2:(3:[])))
```

and its corresponding `DList` looks like this:

```
\x -> (1:(2:(3:x)))
```

i.e., the list is actually a function and the end-of-list `[]` is replaced with the parameter `x`. Once we have a `DList` the only way to ovserve it is to convert it to a list:

```
toList :: DList a -> [a]
toList x = x [] -- remember that x is a function!
```

Question 1. (4pts) Create an empty DList:

```
> toList empty
[]

empty :: DList a
empty =
```

Question 2. (4pts) Create a DList with a single element:

```
> toList (singleton 1)
[1]

singleton :: a -> DList a
singleton x =
```

Question 3. (4pts) Append a `DList` at the back of another:

```
> toList (append (singleton 1) (singleton 2))
[1,2]

append :: DList a -> DList a -> DList a
append xs ys =
```

Question 4. (4pts) Construct a `DList` by "consing" a head element to a tail `DList`:

```
> toList (cons 1 (singleton 2))
[1,2]

cons :: a -> DList a -> DList a
cons x xs =
```

Question 5. (4pts) Convert a regular list into a `DList`:

```
> toList (fromList [1,2,3])
[1,2,3]

fromList :: [a] -> DList a
fromList xs =
```