COMPUTER SCIENCE 312 (Winter Term 2021)
Programming Language Design
Prof. Levy

Problem Set 6

Due on github 11:59pm Friday 16 April

1 Reading Assignment: *Essentials of Programming Languages*, Chapter 4, Sections 4.1,4.2,4.4; Class notes on Haskell

2 Written Problem

Exercise 4.16, page 160 (short answer – just list the component types and the type of the whole expression, not a big type derivation like the ones in the book). Put your answers in a file `ps6.txt` in your github repository.

3 Coding Mergesort in Elegant Python

As we've discussed, a major argument in favor of functional languages like Haskell and OCAML is that they allow you to code up the same algorithm in less code than more popular languages like Python and Java. In this exercise we will put that claim to the test.

As usual, it's a lot less hassle to take something that we already know (Python) and convert it to something we don't yet know (Haskell). So first we're going to download and test a working version of mergesort. Grab it from `stackabuse.com/merge-sort-in-python/`, save it in a file **mergesort.py**, and run it.

As you can see, mergesort looks terribly long and complicated in Python, lending support to the argument in favor of Haskell. If you think about it, though, a lot of the apparently complexity of the Python code you downloaded comes from the use of indices to mutate the list, instead of a functional approach that would return a new list built from splitting and merging. Thanks to the multi-paradigm nature of Python, we can write a version of mergesort that works like this.

To get started, create a new Python file `fmergesort.py` and start it off by writing a function **merge()** that takes two ordered lists and merges them into a big ordered list. As usual with Python, the pseudocode below can be translated line-for-line into Python:

```
MERGE(L1, L2)
  If L1 is empty, return L2
  If L2 is empty, return L1
  If first(L1) < first(L2), return append(first(L1), merge(rest(L1), L2))
  Otherwise, return append(first(L2), merge(L1, rest(L2)))
```

As you may recall, the other import function in mergesort is **split()**, which takes a list and splits it into two lists, by the familiar "round robin" approach (like dealing a deck of cards to two players). Again, if you follow the pseudocode, your Python implementation will be very simple:

```
SPLIT(L1)
  If L is empty, return a pair of empty lists
  If L has just one element, return a pair of lists, the first with that element, and
    the second empty
  Otherwise:
    Recur on the list starting at the third element, yielding lists L1, L2
    Return a pair of lists: append(first(L), L1), append(second(L), L2)
```

Once more, make sure your new **split** function is correct by testing it on a simple list like `[1,2,3,4,5,6]`

Finally, here's the actual Mergesort algorithm, combining the two algorithms you've just implemented:

```
MERGESORT(L)
  If L is empty, return the empty list
  If L contains just one item, return L
  Otherwise:
    Split L into L1, L2
    Call MERGESORT separately on L1 and L2
    Merge the resulting two lists
```

Once you've got mergesort working on the data in the original example, upload your **functional_mergesort.py** to Github.

4 Coding Mergesort in Haskell

If you haven't already installed Haskell on your computer, do it now:
`https://www.haskell.org/platform/`.

Onward to the main event! To get started, here's a little Haskell program **mergesort.hs** that you can copy/paste into a text editor:

```
{--
  mergesort.hs : Mergesort in Haskell

  Doesn't work yet, but it at least compiles and outputs some kind of result!
--}

-- Replace myLength with your actual functions
myLength :: [a] -> Int
myLength [] = 0
myLength (x:xs) = 1 + myLength xs

main :: IO ()
main = print(myLength([33, 42, 9, 37, 8, 47, 5, 29, 49, 31, 4, 48, 16, 22, 26]))
```

Once you've saved this program, compile and run it:

```
ghc mergesort.hs
./mergesort
```

To convert this program to do the actual quicksort, follow the one-function-at-a-time approach you took in the previous section with Python.

5 What to submit

- **Exercise4_16.pdf**
- **fmergesort.py**
- **mergesort.hs**