Assignment 5

CS|E 141 & Inf4mtx 101: Programming Languages

Summer Session II 2016

Due in e3 by September 3rd in the `1X1:Assignment5` dropbox

Place your answers to the following questions in a file named, **`assignment5.hs`**

Question 1:

Write a Haskell function called insert that takes two parameters (a list and a number). You may assume that the list consists of 0 or more numbers that are in order. The function should return a list that would result from inserting the number into the list into the appropriate place to keep the list in order. For example,

`>insert [2, 4, 6, 8]  5` should return `[2, 4, 5, 6, 8].`

Question 2:

Write a polymorphic Haskell function called `insertSort` that takes one parameter (a list of elements). The function should return the list that would result from sorting the list using insertion sort. Note: the insert function should be useful here, as would an auxiliary function.

Question 3:

Write a function `merge :: (Ord a) => [[a]] -> [a]` that takes a finite list of sorted finite lists and merges them into a single sorted list. A "sorted list" means a list sorted in increasing order (using <); you may assume that the sorted lists are finite. For example

```
merge [[1, 2, 3]] = [1, 2, 3]
merge [[1, 3, 5, 7], [2, 4, 6]] = [1, 2, 3, 4, 5, 6, 7]
merge [[1,3,5,7], [2,4,6], [3,5,9,10,11,12]] =
[1,2,3,3,4,5,5,6,7,9,10,11,12]
take 8 (merge [[1, 3, 5, 7], [1,2,3,4,5,6,7,8]]) = [1, 1, 2,
3, 3, 4, 5, 5]
```

Question 4:

Write and test the definition of a (polymorphic) Haskell function '`center`' that takes three arguments, a list arg1 of type `[a]`, a width `arg2` of type `Int`, and a fill item arg3 of type `a`, and returns a list of length `arg2` of type `[a]` containing list `arg1` centered within fill items (i.e., the difference between the number of items preceding arg1 and those following arg1 is at most 1). For instance, `center "abcd" 7 '-'` could yield `"--abcd-"` or `"-abcd--"` (as you choose).

Question 5:

Write and test the definition of a Haskell function 'largest', which finds the largest element of a list, but is implemented using higher-order functions and/or operator sections as appropriate.