CSIT 313 Spring 2019
Dr. Benham

# Haskell Programming Assignment

This assignment asks you to implement three functions in Haskell

1. Implement the **Merge Sort** algorithm as a Haskell function. The function should take a list of values in the Haskell class **Ord** and return a sorted list. The type signature will be

   ```
   mergeSort::(Ord a) => [a] -> [a]
   ```

   Recall that merge sort works as follows:

   a. Split the list into two approximately equal sublists
   b. Sort each sublist using merge sort
   c. Merge the two sorted sublists.

   I am providing some Java code for this algorithm. Your task is to convert this code into Haskell. Note that I made **merge** a private method, since it is only used in mergeSort. For the same reason, you should implement **merge** in Haskell using either the **where** or the **let** construct. (Of course, your implementation will be recursive instead of iterative. As a hint, you will need three base cases: merging two empty lists, merging an empty list with a non-empty list, and merging a non-empty list with an empty list.)

2. In class, we used denotational semantics to evaluate a decimal number using a function $M_{dec}$. We could define a similar function to evaluate *hexadecimal* numbers as follows:

   $M_{hexDigit}$('0') = 0,…, $M_{hexDigit}$('9') = 9,
   $M_{hexDigit}$('A') = $M_{hexDigit}$('a') = 10,…,$M_{hexDigit}$('B') = $M_{hexDigit}$('b') = 11,…,
   $M_{hexDigit}$('F') = $M_{hexDigit}$('f') = 15
   $M_{hex}$("d") = $M_{hexDigit}$('d') (Base case – one-digit string)
   $M_{hex}$("$d_n d_{n-1} \ldots d_1 d_0$") = $M_{hex}$("$d_n d_{n-1} \ldots d_1$")*16 + $M_{hexDigit}$('$d_0$')

   Note that I am allowing both upper- and lower-case letters for the hexadecimal digits for 10,11,…,15.
   Implement a Haskell function mHexDigit to return the numerical value of a single hexadecimal digit. You will need to use guards and the **ord** function from the Haskell module **Data.Char**. The Haskell type should be **mHexDigit::Char -> Int**
   Now, implement a Haskell function **mHex::String -> Int** to evaluate a hexadecimal number given as a string.

   Finally, since mHexDigit is only used by mHex, implement a Haskell function **mHex'::String -> Int** in which mHexDigit is defined using the **where** or **let** contruct.

3. Implement a Haskell function **varmap::String -> [(String,Int)] -> Int**, based on the denotiational-semantics function VARMAP, that takes an identifier and a list of identifier-value pairs and returns the value associated with the identifier in the list. To accommodate Haskell's static typing, you may assume the values are non-negative integers and use -1 in place of **undef**. You may also return -1 as an error code if the identifier is not in the list.

I am providing some sample runs using the GHCiWin of my implementations of these functions so that you can see how they should work in action (and test your implementation).

I am also providing some Haskell functions (together with a sample run) that illustrate the functions **ord**, **chr**, **fst**, and **snd** that you may find useful.