Prerequisites: Basic Functional Programming in Haskell

Liang-Ting Chen

The first part of the following questions requires basic understanding of how to define a function in Haskell. The second part is a recursive function using techniques you should have learned from the first 4 chapters of *Learn You a Haskell for Great Good!*.

Please check your answer using the interactive interpreter `ghci` before submission. If you are not familiar with the command-line interface, please read the first chapter of *Real World Haskell*.

1) Define a function called `myFst` which takes a tuple and returns the first component.

```
myFst :: (a, b) -> a
myFst = undefined
```

2) Define a function `myOdd` which determines if the input is an odd number or not. Hint: You may use `mod` (what is this?).

```
myOdd :: Int -> Bool
myOdd = undefined
```

3) Consider the following function.

```
qs :: Ord a => [a] -> [a]
qs []     = []
qs (x:xs) = qs ys ++ [x] ++ qs zs
  where
    ys = [ y | y <- xs, y <= x ]
    zs = [ z | z <- xs, x < z  ]
```

Please answer the following questions concisely either in plain English or Chinese.

(a) What is `Ord`? What does the type of `qs` mean?

(b) What is the type of `(++)`? What does it do?

(c) What are the elements of `ys` and `zs`, respectively?

(d) What does the function `qs` do? Hint: If you are not familiar with recursive functions (functions which are defined in terms of themselves), run `qs` on some lists (e.g., `[2, 1, 4, 3, 5]`) and make a guess.

(e) Please re-write the function `qs` above and call it `qs'` using **let** expression and **case** expression instead of **where** clause and pattern matching.