CSE341 Section3:  
April 12th, 2018  

Warm-up:  
Write a Haskell function to find the value of the quadratic expression $ax^2 + bx + c$, where a, b, c, and x are any arbitrary Doubles. What is the type of this function?  
(Challenge: what if a, b, c, and x were passed as a single tuple?)

Q1:  
Write a Haskell function to reverse a list. What is the type of this function?  
(Challenges: what if the function reversed the doubled value of the list? That is [1,2,3] -> [6,4,2]. Also, what if the function were tail recursive?)
Q2:
Consider the following Haskell function definitions.

```haskell
alan_example (x:xs) = "something aquatic"
isOdd x = elem x [1,3..x]
slope (x1,y1) (x2,y2) = (y2 - y1) / (x2 - x1)
my_all p y =
    case y of
        [] -> True
        (x:xs) -> p x && my_all p xs
```

Below are a list of possible types for each Haskell function. Next to each one, indicate if it is V (valid) or IV (invalid). That is, if you were to add these as type declarations, would it compile?

- `alan_example :: [a] -> [Char]` V
- `alan_example :: (a) -> [Char]` IV
- `alan_example :: [Integer] -> [Char]` V

- `isOdd :: Integer -> Bool` V
- `isOdd :: Double -> a` IV
- `isOdd :: a -> Bool` IV

- `slope :: (Integer, Integer) -> (Integer, Integer) -> Double` V
- `slope :: (Double, Double) -> (Double, Double) -> Double` V
- `slope :: (Integer, Double) -> (Integer, Double) -> Double` V

- `my_all :: (a -> Bool) -> [a] -> Bool` V
- `my_all :: (Integer -> Bool) -> [Integer] -> Bool` V
- `my_all :: (Integer -> a) -> [Integer] -> Bool` V

Q3:
Write a function `my_map2` that is analogous to `map` but works for functions of two, equally long arguments rather than one. What is its type? For example, “`my_map2 (+) [1,2,3] [4,5,6]`” should evaluate to “[5,7,9]”. (Challenges: write another function “double” that uses `my_map2`. Also, extend `my_map2` to work for any length arguments by choosing the length of the smaller list as the result).