Mark Gameng, A20419026

CS 440, James Sasaki

HW 01

1. In ghci, what happens on the inputs below? Give results or briefly describe the errors
    a. Sin (cos pi)
       -0.8414709848078965
    b. Cos -1
       Error because in haskell, there are no negative constants. -1 is a function call of unary – on the integer 1. In this case, it is basically doing (cos -)1, so it results in an error because it is doing the cosine of just the function – . If added a parenthesis, cos (-1) it would work.
    c. Sin cos pi
       Error because of the ordering from left to right. Basically, with parenthesis, it does this sin(cos) pi. So, the error is because it is trying to do a sine of just the cosine function with no arguments. It would work if it was sin(cos pi) since cos pi evaluates to a number.
    d. (sqrt . head [sqrt]) 16.0
       2.0
2. What do you get if you delete all the extra and the problematic parentheses from the expressions below?
    a. (cos(sqrt(2.5))+((sin)(pi)))(*)(2)
       cos(sqrt 2.5) + sin pi * 2
       Result of doing this in ghci: -1.0342318905208982e-2
    b. ((:) ((‘a’ : (“b”)) ++ “cd”)) (([([‘c]) ++ “(d)”]))
       (:) ('a' : "b" ++ "cd") ([['c'] ++ "(d)"])
       Result of doing this in ghci : ["abcd","c(d)"]
    c. ([([([17])])] : ([([ ])]))
       [[[17]]]:[[]]
       Result of doing this in ghci: [[[[17]]],[]]
3. Rewrite the expression below so that it uses prefix functions throughout
    a. ((a + b) * c) / (d ^e)
       (/)((*)((+)a b) c)((^)d e)
       Result of ((a + b) * c) / (d ^e) with a = 2, b = 3, c = 4, d = 5, e = 6: 1.28E-3
       Result of (/)((*)((+)a b) c)((^)d e) with a = 2, b = 3, c = 4, d = 5, e = 6: 1.28E-3
4. Rewrite the following expression so that it uses infix notation throughout
    a. F (g x (h a b)) (c (d e f))
       f * (g `x` (h `a` b)) * (c (d `e` f))
       Using:
       x x y = 2 * x * y   a x y = x^2 / y^3   e x y = sqrt(x + y)   c x = log x
       f = 2, g = 3, h = 4, b = 5, d = 6
       Result of doing f * (g `x` (h `a` b)) * (c (d `e` f)) in ghci: 1.597
       Tried to use as much infix notation on a function as much as possible.

5. Complete the following function definition so that on any list, f returns True

    a. F x = x == [x !! i ???]
    f x = x == [x !! i | i <- [0 .. length x – 1]]
    Result of doing f []: true
    f [3,9,0]: true
    f ['2','3']: true

6. Complete the following function definition: stutter n x should return a list of length n where each element is x.

    a. Stutter n x = [???1 | ???2 <- ???3]
    stutter n x = [x | n <- [1 .. n]]
    Result of stutter 3 5: [5,5,5]
    Result of stutter 6,7: [7,7,7,7,7,7]
    Result of stutter 0, 5: []
    The values of ???3 don't matter but the length of the list does. One example is it could also be [2 .. n+1] and it would still be correct. But, it can't be [1 .. n+1] or [2 .. n] because it would result in an extra element or one less respectively.

7. Let g be the list defined below ….

    a. rot x = last x : init x
    g = [1,3,5] : [rot x | x <- g]
    Some properties:
    (1) take (m+1) g = head g: take m(tail g)
    (2) take (m+1)g = take m g ++ [e]
    for take n g, n = 0, 1, 2, …
    take 0 g = []
    take 1 g = [1,3,5] : [rot x | x <- []] = [1,3,5] : [] = [[1,3,5]]
    That is because of property (1)
    take 2 g = [1,3,5] : [rot x | x <- take 1 g] = [1,3,5] : [rot x | x <- [[1,3,5]]]
    = [1,3,5] : rot [1,3,5] : [rot x | x <- []] = [1,3,5] : [5,1,3] : [] = [[1,3,5],[5,1,3]]
    take 3 g = take 2 g ++ [e]
    Using property (2), and [e] is just the expression for the last element of take(m+1)g which would just be rot of the last element of take m g
    Thus, take 3 g = take 2 g ++ [e] = [[1,3,5],[5,1,3]] ++ [rot [5,1,3]] = [[1,3,5],[5,1,3]] ++ [3,5,1] = [[1,3,5],[5,1,3],[3,5,1]]
    take 4 g = take 3 g ++ [e] = take 3 g ++ [rot[3,5,1]] = [[1,3,5],[5,1,3],[3,5,1]] ++ [1,5,3] = [[1,3,5],[5,1,3],[3,5,1],[1,3,5]]
    As you can see, the pattern is that the next element is just the rot of the previous element, except when it's for the zeroth index. For example, [5,1,3] is rot[1,3,5]; in this case, [5,1,3] is index 1, and [1,3,5] is index 0. The next element for take 4 g would then be rot[1,3,5] or [5,1,3]. So take 5 g = take 4 g ++ [[5,1,3]] = [[1,3,5],[5,1,3],[3,5,1],[1,3,5],[5,1,3]]
    This can be written as:
    For m > 0,
    take (m+1) g = take m g ++ [rot(take m g !! (length(take m g) – 1))]