2018-01-17 Tutorial

Haskell

```haskell
data [t] = []
         | t : [t]

prepend :: Integer -> [[Integer]] -> [[Integer]]
prepend i [] = []
prepend i (x:xs) = (i:x) : y
    where y = prepend i xs

insert :: Integer -> [Integer] -> [[Integer]]
insert i [] = [[i]]
insert i y@(x:xs) = (i:y):z
    where y = insert i xs
          z = prepend x ys

add :: Integer -> [[Integer]] -> [[Integer]]
add _ [] = []
add i (x:xs) = a ++ b
    let a = insert i x
        b = add i xs
    in a ++ b

mystery :: [Integer] -> [[Integer]]
mystery [] = [[]]
mystery (x:xs) = add x (mystery xs)

comb :: [Integer] -> Integer -> [[Integer]]
comb y i
    | i == 0    = [[]]
    | y == []   = []
    | otherwise = a ++ c
        where a = comb (tail y) i
              b = comb (tail y) (i - 1)
              c = prepend (head y) b

-- key search
[("Alice", (1414, "cafebabe")),
 ("Bob", (2718, "deadbeef")),

 ("Eve", (3141, "baddfood"))]
```

```
39
40  data Maybe = Just t2
41             | Nothing
42
43  lookup :: t1 -> [(t1, t2)] -> Maybe t2
44
45  -- first solution: use pattern matching
46  lookupX :: String -> [(String, (Integer, String)] -> (Integer, String)
47  lookupX a b =
48     case lookup a b of
49         data v -> v
50         Nothing -> error "bad"
51
52 lookup a b =
53     fromMaybe "Empty" (lookup a b)
```

- What is y@(x:xs)??

```
1  $ /u/cs146/pub/marmoset_submit cs146 Q1 Q1.rkt
```

```
1  [1..100]
2  [2,4,..100]
3  [x|x=[1..100],mod x 13 == 0]
4
5  [(x,y,z)|x <- [1..100], y <- [x..100], z <- [y..100], x^2+y^2==z^y]
6
```