FUNCTIONAL PROGRAMMING 2017 MOISE GABRIEL

QUESTION 1

(a)

```
> (.) :: (b -> c) -> (a -> b) -> a -> c
> f.g = \x -> f (g x)

> zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]
> zipWith _ [] _ = []
> zipWith _ _ [] = []
> zipWith f (x:xs) (y:ys) = f x y : zipWith f xs ys

> uncurry :: (a -> b -> c) -> (a, b) -> c
> uncurry f (x,y) = f x y
```

(b)

```
filter p . map f = map f . filter (p . f)
filter p . concat = concat . map (filter p)
zipWith f = map (uncurry f) . zip
```

(c)

```
> pascal :: [[Integer]]
> pascal = iterate (\xs -> zipWith (+) (0 : xs) (xs ++ [0])) [1]
```

(d) `> f g h x y = g . h x . g y`

From the definition of (.), we know that

(h x) :: b -> c => h :: x -> b -> c

(g y) :: a -> b => g :: y -> a -> b

(h x) . (g y) :: a -> c

From the same definition, we have

g :: c -> d

=> y = c and d = a -> b

g . (h x) . (g y) :: a -> d => f g h x y :: a -> d

Therefore, we have

f :: (c -> a -> b) -> (x -> b -> c) -> x -> c -> a -> a -> b.