Lawrence Allen
Assignment 1

1.1.

a)

Code:

```
#!/usr/bin/env stack
-- stack --install-ghc runghc
allEven :: [Int] -> Bool
allEven [] = True
allEven(x:xs) = ((x `mod` 2) == 0) && allEven(xs)

main = do
    print (allEven [1,3,5,8,6])
    print (allEven [2])
    print (allEven[2,4,6,0])
```

Output:

```
False
True
True
```

b)

Code:

```
#!/usr/bin/env stack
-- stack --install-ghc runghc

f :: [Int] -> [Int]
f(x) = zipWith(*) [0..length(x)] x

main = do
    print (f[3,5,7])
    print (f[])
    print (f[1])
```

Output:

```
[0,5,14]
[]
[0]
```

1.2.

a)

Code:

```
#!/usr/bin/env stack
-- stack --install-ghc runghc
-- | comment
quadsolns :: Float -> Float -> Float -> [Float]
quadsolns a b c =
        if ((b * b) - (4 * a * c)) < 0
          then []
        else
          [(-b + sqrt((b * b) - (4 * a * c)))/(2*a) , (-b - sqrt((b * b) - (4 * a * c)))/(2*a)  ]

main = do
      print (quadsolns 1 2 (-4))
      print (quadsolns 1 4 10)
      print (quadsolns 2 (-5) 3)
```

Output:

```
[1.236068,-3.236068]
[]
[1.5,1.0]
```

b) Roots are 1.5 and 1

1.3.

a)

Code:

```
#!/usr/bin/env stack
-- stack --install-ghc runghc
-- | comment
isPrime :: Int -> Bool
isPrime 1 = False
isPrime x = x > 1 && length(filter(==0) (map (x `mod` ) [2..(x-1)])) == 0

main = do
      print (isPrime 3)
      print (isPrime 7)
      print (isPrime 8)
      print (isPrime (-2))
      print (isPrime 1)
      print (filter(isPrime) [1..30])
      let allPrimes = filter(isPrime) [1..]
      print "Done!"
```

Output:

```
True
True
False
False
False
[2,3,5,7,11,13,17,19,23,29]
"Done!"
```

b) [2,3,5,7,11,13,17,19,23,29]

c) let allPrimes = filter(isPrime) [1..]

d) Haskell has lazy evaluation. As long as I never do anything with allPrimes, it’ll never evaluate it, so it won’t get in an infinite loop.