ECS 140A Homework 5 – Problem 1

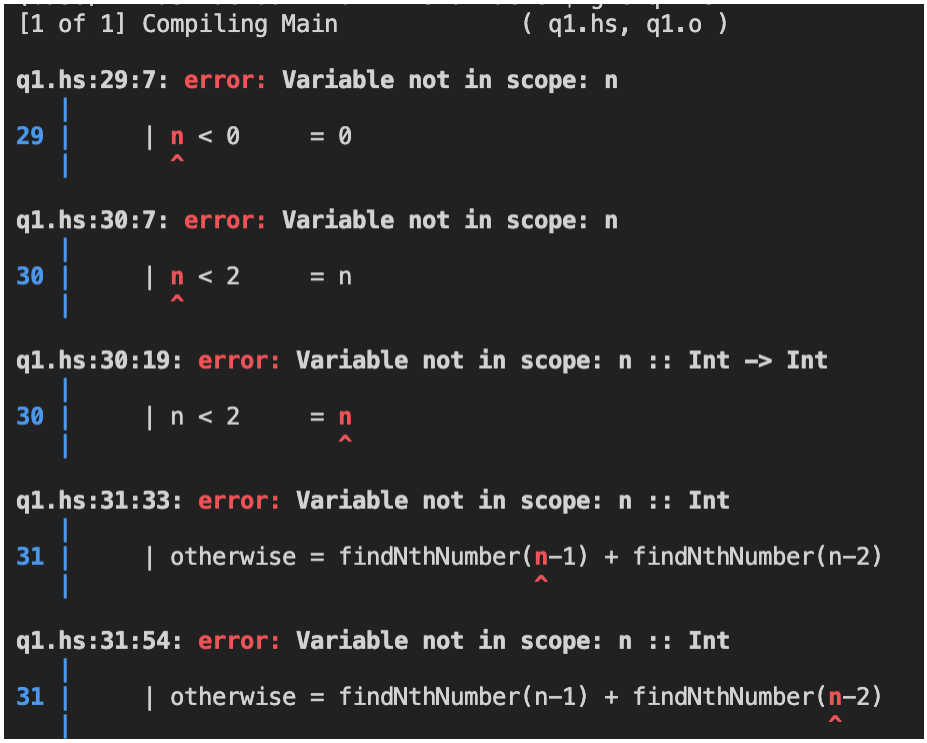
1 Haskell

Step 1: Algorithm/Pseudocode

```
findNthNumber(Int) -> Int
    if n < 0 return 0
    else if n <= 1 return n
    else return findNthNumber(n-1) + findNthNumber(n-2)

fibSeq(Int) -> List
    use map function with findNthNumber applying from 0 to n,
    automatically compiling into a list to return
```

Step 2: Actual Code

```
findNthNumber :: Int -> Int
findNthNumber
    | n < 0    = 0
    | n < 2    = n
    | otherwise = findNthNumber(n-1) + findNthNumber(n-2)

fibSeq :: Int -> [Int]
fibSeq n = map findNthNumber [0..n]

main = do
    print (fibSeq 0)
    print (fibSeq 1)
    print (fibSeq 9)
```



This error was raised because I forgot to define `n` as the argument passed to the function `findNthNumber`. To fix this, I simply added the argument in to the function signature.

Incorrect code:

```
findNthNumber :: Int -> Int
findNthNumber
    | n < 0    = 0
    | n < 2    = n
    | otherwise = findNthNumber(n-1) + findNthNumber(n-2)
```

Correct code:

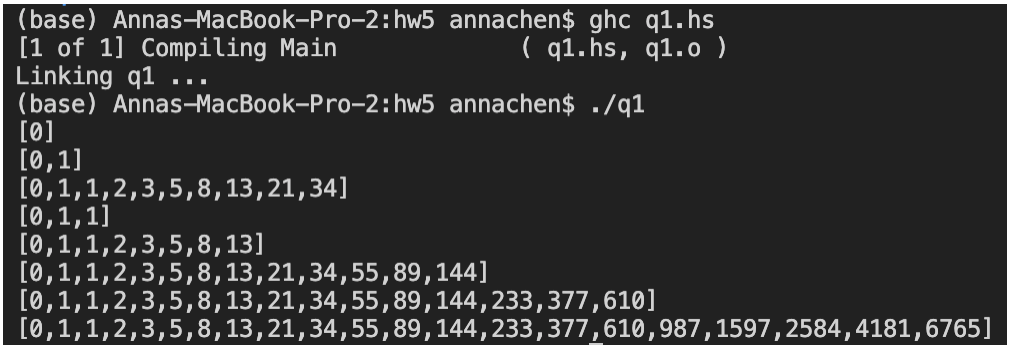
```
findNthNumber :: Int -> Int
findNthNumber n
    | n < 0    = 0
    | n < 2    = n
    | otherwise = findNthNumber(n-1) + findNthNumber(n-2)
```

Step 3: Working Code

```haskell
findNthNumber :: Int -> Int
findNthNumber n
    | n < 0    = 0
    | n < 2    = n
    | otherwise = findNthNumber(n-1) + findNthNumber(n-2)

fibSeq :: Int -> [Int]
fibSeq n = map findNthNumber [0..n]

main = do
    print (fibSeq 0)
    print (fibSeq 1)
    print (fibSeq 9)
```

Step 4: Debug Process

All given test cases passed and behaved as expected. All additional test cases have also passed and behaved as expected. These test cases can be found at the end of this report. In the following screenshot, note that the first three output lines are the output of the given cases.



Step 5: Add Documentation

```haskell
-- References:
-- https://www.tutorialspoint.com/haskell/haskell_functions.htm

-- Find the nth number in the Fibonacci sequence
findNthNumber :: Int -> Int
findNthNumber n
    | n < 0    = 0 -- if n < 0, return 0
    | n < 2    = n -- if 0 <= n <= 1, return n
    | otherwise = findNthNumber(n-1) + findNthNumber(n-2) -- use recursion to get sum of
        last 2 numbers

fibSeq :: Int -> [Int]
fibSeq n = map findNthNumber [0..n] -- apply findNthNumber to array [0..n]

main = do
    -- given cases
    print (fibSeq 0) -- [0]
    print (fibSeq 1) -- [0, 1]
    print (fibSeq 9) -- [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

    -- additional cases
    print (fibSeq 2) -- [0, 1, 1]
    print (fibSeq 7) -- [0, 1, 1, 2, 3, 5, 8, 13]
    print (fibSeq 12) -- [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
    print (fibSeq 15) -- [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
    print (fibSeq 20) -- [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987,
        1597, 2584, 4181, 6765]
```

Step 6: Extra Test Cases Used

- `fibSeq 2` – [0, 1, 1]
- `fibSeq 7` – [0, 1, 1, 2, 3, 5, 8, 13]
- `fibSeq 12` – [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
- `fibSeq 15` – [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
- `fibSeq 20` – [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765]