quiz 1 solutions

1. Answer as shown with the example from class:

   `https://webdocs.cs.ualberta.ca/~hayward/304/jem/warmup.html#sqrt`

```
  --------          --------          --------
\/ 4 70 93 = ?    \/ 4 54 19 = ?     \/ 4 71 71 = ?

*** step 1: largest x such that x*x <= 4?   x = 2

   2                 2                  2
  --------          --------           --------
\/ 4 70 93        \/ 4 54 19         \/ 4 71 71
   4                 4                  4
   -------           -------           --------
     70                54                 71

*** step 2: largest y such that (2*20*2 +y)*y <= 70? 54? 71?

....
```

   Check the rest of your answer using `old_sqrt.py` from the github repo.

```
4 = 2 * 2 + 0
70 = 41 * 1 + 29
2993 = 427 * 7 + 4   47093 = 217 * 217 + 4
-----------------
4 = 2 * 2 + 0
54 = 41 * 1 + 13
1319 = 423 * 3 + 50  45419 = 213 * 213 + 50
-----------------
4 = 2 * 2 + 0
71 = 41 * 1 + 30
3071 = 427 * 7 + 82  47171 = 217 * 217 + 82
```

2. a) Check your answer by executing the python code.

   b) Proof. Argue by contradiction. Let x be the smallest integer for which `collatz(n)` does not terminate. Assume that x is even, say $x = 2 \times y$. Inputs to `collatz` are positive, so we can assume that x is at least 2, so y is at least 1.

   `collatz(x)` initializes parameter n with value x and then enters the while loop. Case 1: $n = 1$. Execution skips the loop body.

   Case 2: $n \neq 1$. Execution enters the loop body. Since n is even, n is now assigned the value `n//2`, so the value of n is now y, and execution continues. Since $y < x$, by assumption `collatz(y)` terminates, so the while loop execution terminates.

   In each case the original execution of `collatz(x)` terminates, contradicting our assumption. So our assumption must be false. So our assumption that x is even must be false. So if the Collatz conjecture fails for some integer, and x is the smallest such integer, then x cannot be even, so x must be odd.

3. Consider a call to `ifib(n)`. Here are the operations performed:

```
a,b = 0,1
a,b = b, a+b    (for loop, _ is 0)
a,b = b, a+b    (for loop, _ is 1)
a,b = b, a+b    ...
a,b = b, a+b    (for loop, _ is n-1)
```

In each line we have the assignment `a=b`, the addition `a+b`, and the asssignment `b=a+b`. For each of these three operations, the time needed is proportional respectively to the number of bits in `b`, the number of bits in the sum `a+b`, and the number of bits in the sum `a+b`, so in time $\Theta(\lg(b) + \lg(a+b) + \lg(a+b) = \Theta(\lg(a+b))$.

The number of bits needed to store an integer k is less than $1+\lg(k)$ so in $\Theta(\lg(k))$.

The j'th time the loop executes, $a + b = \text{fib}(j + 2)$,

so the total runtime is in

$$\Theta(1) + \Theta(\sum_{j=0}^{n-1} \lg(\text{fib}(j+2))) =$$

$$\Theta(\sum_{j=1}^{n} \lg(\text{fib}(j))) + \Theta(\lg(\text{fib}(n+1))) =$$

$$\Theta(\sum_{j=1}^{n} \lg(\text{fib}(j)))$$

since $\lg(\text{fib}(n+1))$ s in $\Theta(\lg(\text{fib}(n)))$.