Do problems 6.17 **(6 points)** and 6.18 **(6 points)** in the textbook. Also, run the following experiment: implement the Rabin-Miller algorithm where the input is assumed to be an integer given in binary (see Problem 6.11) – you may use the implementation given here **(4 points)**. Now, for as many inputs $(n)_b$, that is number n given in binary, if n is not prime, computer how many witnesses (of compositeness) are there. Plot the result, and hypothesize on a good asymptotic approximation to the function $f_w(n) = m$ where m is the number of witnesses for a given n (of course, $m = 0$ if n is prime) **(4 points)**.

(i) **(6 points) Problem 6.17.** *Consider Shank's algorithm—algorithm 6.4. Show that Shank's algorithm computes x, such that $g^x \equiv_p h$, in time $O(n \log n)$ that is, in time $O(\sqrt{p} \log(\sqrt{p}))$.*

The algorithm runs in $O(n \log n)$ because the computation of $L1$ and $L2$ each perform n modular exponentiations. Repeated squaring to perform a single modular exponentiation runs in O(log n) time, thus making the final time complexity $O(n \log n)$.

Note: The intersection can be performed in O(n) time if $L1$ and $L2$ are both implemented as hashed sets (dictionaries in Python) giving them O(1) lookup time.

(ii) **(6 points) Problem 6.18.** *Implement algorithm 6.4.*

An example implantation can be seen at:
https://github.com/michaelsoltys/IAA-Code/tree/master/Problems/6.18/Solution-1

(iii) **(4 Points)** *Implement the Rabin-Miller algorithm where the input is assumed to be an integer given in binary.*

An example implantation can be seen at:
https://github.com/michaelsoltys/IAA-Code/tree/master/Problems/6.11/Solution-1

(iv) **(4 Points)** *Plot the result, and hypothesize on a good asymptotic approximation to the function $f_w(n) = m$ where m is the number of witnesses for a given n (of course, $m = 0$ if n is prime).*

A good approximation of the upper bound is n and $\frac{3n}{4}$ for the lower bound.