A. In the (non-obscure, readable) programming language of your choice, implement the continued fractions approximation algorithm described in class. I wrote mine as a Python function. Here is a precise specification. The inputs are `x0` and `d` such that:

1. `x0` is a floating-point number such that $0 \leq x_0 < 1$.
2. `d` is a non-negative integer — the "depth", meaning that the approximation truncates after a_d by assuming that $x_{d+1} = 0$.

The output is a pair (array, list, struct, object, etc.) `[a, b]` such that

1. `a` is a non-negative integer.
2. `b` is a positive integer, such that $\gcd(a, b) = 1$ and $a/b \approx x_0$.

This paragraph is not part of the assignment, but: You might want to test your code (to varying depths) against the two examples in Mermin's Appendix K. You also might want to think about how you would prove that your output satisfies $\gcd(a, b) = 1$. My proof is easy.

B. What happens if you run your code (to varying depths) on input `x0` equal to `0.5`? What about `0.2`? `0.3`? `0.333`? `1.0 / 3.0`? Why? (My code does interesting things. Maybe yours will too, or maybe not.)