II-1. Let $n = 2^m + 1$ with $m \geq 2$.

(i) Show that

$$n \quad \text{is prime} \qquad \Longleftrightarrow \qquad 3^{\frac{n-1}{2}} \equiv -1 \bmod n.$$

(ii) Show that the problem of deciding whether or not n (of the above form) is prime is in P.

(iii) Prove: n prime $\Rightarrow$ m even.

(iv) Write $m = 2^k r$, with r odd. Find a non-trivial factorization of n if $r > 1$.

(v) Give an alternative encoding for the problem of deciding whether or not n of the given form is prime that makes the test in part (i) exponential instead of polynomial.

II-2. [**compositeness test**] Implement the Miller-Rabin probabilistic compositeness test, as describes on pages 27–28 of the Chapter on 'Four Number Theoretic Problems'. Your functions should take as input a positive odd integer n to be tested, as well as a positive integer k that signifies the number of attempts to find a witness for the compositeness test before n is declared 'probably prime'. The output should consist of either a witness and the declaration 'n is composite' or the declaration 'n is probably prime since it passed k compositeness tests'.

II-3. [**prime certificate**] Implement an algorithm that generates a certificate for the primality of an odd prime number n, by finding an integer a for which $a_i^{\frac{n-1}{2}} \equiv -1 \bmod n$, and for every odd prime divisor p_i of $n-1$ an integer a_i satisfying $a_i^{\frac{n-1}{p_i}} \not\equiv 1 \bmod n$, and then recursively applying this to the odd primes p_i.

II-4. [**Pollard-ρ**] Implement Pollard's ρ algorithm, for integer factorization. Try to speed it up as much as you can. As an indication of its performance, if m is a product of two primes of k and $2k$ decimal digits, describe approximately how the running time varies as a function of k.

II-5 Combine the previous algorithms into one function that, on input a positive integer n, returns the complete factorization of n together with primality certificates for each of the odd prime factors.