Solution T 6.2 $\implies$: part a) is stated as a theorem in the notes. As for part b): by irreducibility this gcd is either 1 or f. If it is f then f divides $x^{p^{n/\ell}} - x$, but the latter factors into irreducible polynomials of degree dividing n/ℓ, but f has degree n, so that f cannot be a divisor of $x^{p^{n/\ell}} - x$.

$\impliedby$: By a), f is square-free (separable) and is a product of irreducible polynomials of degree $\mid n$. If f had a factor g of degree $< n$, then $\deg(g) \mid \frac{n}{\ell}$ for some prime ℓ. Then we would have $g \mid \gcd(f(x), x^{p^{n/\ell}} - x)$, contradicting b). Thus, f is a product of irreducible polynomials of degree exactly n. Since f has degree n itself, it follows that f must be irreducible.

Solution P 6.3 Below we give an algorithm which, given a square-free polynomial $f \in \mathbb{F}_q[t]$ and an integer $d \geqslant 1$, computes the product g_d of all irreducible factors of f of degree exactly d.

The key point to compute $\gcd(f, t^{q^d} - t)$ fast is to just compute $t^q, t^{q^2}, \ldots \pmod f$ iteratively (just use Euclidean division), and then compute $\gcd(f,\ \ (t^{q^d} \bmod f) - t)$.

```
def exponentiation_mod(P, m, f):
    """
    Given two polynomials f, P in F_q[X] and an integer m > 0, we compute
        P^(q^m) mod f via fast exponentiation (iterated Frobenius)
    """
    if m == 1:
        return (P^q).quo_rem(f)[1]
    P1 = exponentiation_mod(P, m - 1, f)
    return (P1^q).quo_rem(f)[1]


#Here we assume that f is given as a square-free element of F_q[t].
def factor_fixed_given_degree(f, d, q, r=1):
    R.<t> = PolynomialRing(GF(q))
    g = gcd(f, derivative(f))
    assert(g == 1) #avoid f not-squarefree
    tqr = exponentiation_mod(t, r, f)
    f_r = gcd(f, tqr - t)

    if r == d:
        return f_r
    f = R(f / f_r)
    r += 1
    return factor_fixed_given_degree(f, d, q, r)
```