Abstract: A basic precept of symbolic computation is to compute the exact answer to a problem whatever the cost; this is in stark contrast to so-called "numerical computation" where approximate answers are the norm and speed is especially important. A well-known phenomenon in computer algebra is "intermediate expression swell" (also called "coefficient explosion"): this is said to happen when a computation on small inputs produces a small result but some of the intermediate results are very much larger (leading to unreasonably long computation times). A widely used device for mitigating the problem is to use a modular method: instead of computing with rational numbers, we perform the computation over several finite fields, and then reconstruct the desired result.

Before adopting a modular approach, one must investigate the issue of "bad reduction", and how to handle it. Bad reduction occurs when the result of the modular computation is different from the modular reduction of the desired result. The appropriate way to handle bad reduction depends on the algorithm being considered.

We start by looking at the fundamentals of a modular approach, including the essential reconstruction phase. Then we see some details of how a modular approach has been actually been applied, including
the corresponding good--bad categories of primes.