Solution notes

Foundations of Computer Science 2005 – Paper 1 Question 1 (LCP, 10 marks)

This question particularly examines Lecture 12, Polynomial Arithmetic, but it also relates to Lectures 1–6, especially list programming, sorting and O-notation.

(*a*) We know (from the lecture notes) that addition of *univariate* polynomials is easy provided polynomials are represented by sorted lists. Then addition is a kind of ordered merge. An equality test simply has to compare the two polynomials term by term, and ML's built-in equality test could be used. The lecture notes used a *sparse* representation, which ignores terms that have a coefficient of zero. The univariate polynomial $a_n x^n + \cdots + a_1 x + a_0$ can be represented by a list of (i, a_i) pairs, for $a_i \neq 0$, sorted by the exponent i.

Coefficients should ideally be rational numbers, represented as pairs of arbitrary-precision integers having no common factor. To keep things simple, we can get away with using floating-point numbers (type `real`), while being aware of the risk of rounding errors.

A similar representation works for the multivariate case. However, we have to store the names of the variables as well as their exponents. A canonical representation—one that maps equal polynomials to equal data structures—makes the equality test trivial and should simplify the implementation of addition. So, in representing a term, we keep the variables in alphabetical order. We obviously omit those with an exponent of zero.

Polynomials need to be sorted somehow. (Otherwise, the equality test will be quadratic, and addition will also be quadratic unless we don't bother to collect like terms.) To sort polynomials requires an ordering on terms such as xy^2 and x^2y. One possibility is to expand them as xyy and xxy and to use the standard dictionary ordering. Many orderings are acceptable; the main thing is that candidates recognize the need for an ordering and make a plausible suggestion.

So, the data structure for polynomials is a list of (p_i, a_i), where a_i is a nonzero coefficient of type `real` (or `rational`) and p_i has type `(string*int)list` and denotes a product of variables raised to various powers. Elements of this list are sorted from largest to smallest using the term ordering.

(*b*) Addition of multivariate polynomials, represented as outlined above, is very similar to the univariate case. If either list is empty then the result is the other list. If both are non-empty, then compare the two leading terms (which will be first, thanks to the ordering). If the two terms are equal, then add their coefficients to obtain the term to include in the result, but it should be omitted if the sum of the coefficients is zero. If the two terms differ, then take the larger of the two as the leading term in the result, and recursively add the remaining polynomials.

As in the univariate case, the time taken to add polynomials of lengths m and n is

$O(m+n)$. This is because each iteration removes a term from at least one of the input polynomials.

As for equality, ML's built in equality test will deliver correct results. The two polynomials are compared term by term, with a result of `false` if any difference is found. The cost is $O(\min(m,n))$: each iteration eliminates a term from each polynomial.

Both O-notation estimates assume that the number of distinct variables is small enough to be taken as constant.