Computer Science II — Homework 6 — Polynomial Class

This assignment is due Thursday, March 15 at 11:59:59pm and is worth 70 points toward your homework grade. You may use any technique you wish from the first half of the semester (through Lecture 11 and Lab 6) to solve this problem.

Your problem is to implement a *Polynomial* class. This class should be able to represent, evaluate, add, subtract and multiply polynomials. The polynomials should have integer coefficients and non-negative integer exponents. For example,

$$f(x) = 5x^{13} - 3x^6 + 4x - 1$$

is a degree-13 polynomial, with coefficients 5, -3 , 4 and -1, and with exponents 13, 6, 1 and 0. The variable x can be any real number. It is assumed that you know enough about polynomials to complete the assignment.

Your class should have the following operations:

- Construct an empty polynomial.
- Construct a polynomial from two equal-length vectors of integers, with the first vector giving the non-zero coefficients and the second vector giving the associated exponents. If the two vectors are of different lengths, only form the terms up to the shorter length. For example, if the input coefficient vector contained 5, -3, 4 and 1 and the input exponential vector contained 13, 6 and 0, then the resulting polynomial should be

  $$f(x) = 5x^{13} - 3x^6 + 4.$$

  Any terms with negative exponents or coefficient values of 0 should be ignored. Finally, if the same exponent appears more than once, the coefficient associated with the last appearance of the exponent should be used. You may not assume the input exponent values are ordered, even though the above example is (and this is the typical case).
- A copy constructor.
- `degree`: Provide the degree of the polynomial. This is the value of the highest exponent with non-zero coefficient.
- `add_term`: Add a term to the polynomial, replacing the previous term with the same exponent if it already exists. If the coefficient is 0 the term should be removed from the polynomial (if it is there). If the exponent is negative, the function should not change the polynomial.
- `add`: Add two polynomials to create a third polynomial.
- `subtract`: Subtract two polynomials to create a third polynomial.
- `multiply`: Multiply two polynomials to create a third polynomial.
- `eval`: Evaluate a polynomial given a particular value of x, returning a real number.
- `write`: Write the polynomial to an output stream. The output should all be on one line, with terms in decreasing order of exponent. See the example posted on-line for the format. This output is not as "pretty" as it might be, but it prevents you from having to worry about too many special cases.

In all cases if the coefficient of a term becomes 0, then it should be removed from the polynomial. As described above, in effect this allows the `add_term` function to act as a "remove-term" function by providing a zero coefficient for the term to be removed. If all terms of a polynomial have a coefficient value of 0 then the value of the polynomial is 0.

A sample main program is available on the web site. You will notice that the syntax of using these functions is somewhat awkward. An alternative is to use operators. A second main program is available which replaces many of these functions with operators. This includes the use of `operator()` and the use of `operator<<`, both of which are syntactically a bit tricky. As a result, there will be **two submission sites available for this homework**. The first site is for use of the first main program. The second site is for use of the second, operator version of the program. If you correctly implement the operator version, including all operators, you will earn an extra 5 points on the homework. You do not need to submit to the first site if you submit to the second (but you may, if you want, to ensure that you have earned at least the basic 60 points). Lecture notes on the use of operators, which will not be not covered in class, will be posted on the course web page.

Final notes:

- You will need to evaluate x^n, where n is an integer. One simple way to do this is the `pow` function (look it up), but there are others.
- Part of your assignment grade will depend on the efficiency of your implementations. For example, if there are N terms with non-zero coefficients in one polynomial and M terms with non-zero coefficients in the second, then the time required for the `add` operation should be at most $O(M + N)$. Unfortunately, the multiplication operation can not be performed in better than $O(MN)$ time and your algorithm is likely to be slower than this.
- You will need to submit three files: `Polynomial.h`, `Polynomial.cpp` and `readme.txt`.