hw08: Linked lists

Focus:

- Linked Lists
- Operator Overloading
- Copy control
- Separate compilation

Problem:

You will implement a class `Polynomial` for storing and manipulating polynomial expressions. A polynomial is an expression of the form

$$a_k x^k + \ldots + a_2 x^2 + a_1 x + a_0$$

The number k is called the **degree of the polynomial** and the numbers a_0, ..., a_k are called the **coefficients**.

Store each polynomial as a degree and a singly linked list of its coefficients, but "backwards", i.e. with the lowest degree coefficient first. (Backwards makes your program easier!) For example, the polynomial $4x^3 + x + 7$ would be stored as:

```
    head_ptr --> 7 --> 1 --> 0 --> 4
    degree:  3
```

Here the constant term, 7, is first, then the linear (x) term, 1, then the quadratic (x^2) term, 0, and finally the highest degree term with a coefficient of 4. In addition, the degree of the polynomial, 3, is stored in the member variable `degree`.

Storing the polynomials in this order, rather than in the order in which we normally write them, will make it easier to perform arithmetic operations.

The class *invariant* (i.e., the thing that must be true about the class whenever a member function is *finished* making changes) is:

1. The coefficients are stored in a linked list with the constant term at the front of the list and the highest order term at the end of the list.
2. The zero polynomial is represented by a list with a single item, zero. The degree of the zero polynomial is 0.

3. The list for a non-zero polynomial does not end with a zero coefficient. (In other words, the highest degree coefficient is only zero if this is the zero polynomial.)
4. The value of the `degree` member variable is one less than the length of the list.

What you have to handle:

- constructor that takes a vector of its coefficients in order from highest degree term to lowest
- +=
- +
- copy control, i.e. destructor, copy constructor and assignment operator
- ==
- !=
- << Use the caret ^ for exponentiation. So: 5x^3 represents five times the term with an exponent of three.
- evaluation. Takes a single argument and evaluates the polynomial for that value of "X".

Notes

- Coefficients can be either ints or doubles, as you prefer.
- Feel free to make your life easier by using the linked list code that we developed. Include the Node struct definition and function prototypes in your polynomial.h and the function implementations in polynomial.cpp.

Test program: attached is a test program. Feel free to expand on it. You may not be familiar with all of the syntax used in this program.

- boolalpha. This *manipulator* tells the output stream that when it is printing a bool value, true or false, to print it that way, i.e. as the strings "true" and "false". Without this manipulator, C++ will print true as 1 and false as 0.
- {2, 4, 8}. The use of braces around a comma-delimited sequence of values can be used to initialize a vector to hold that sequence of values. Please note that this use of braces requires C++11.

What to hand in:

Header and implementation files for polynomial (polynomial.h and polynomial.cpp).