In this assignment, you will implement a `PolynomialLinkedList` class, that stores only the monomial terms with nonzero coefficients. Each node in the list stores a structure consisting of two integer values, the coefficient and the exponent of the corresponding monomial term (e.g., the monomial $15x^2$ would be stored as the pair of integer values 15 and 7). The nodes must be listed in descending order of their exponent values. Your class will require the following functionality:

- A default constructor, a copy constructor, a destructor, and an assignment operator.
- An `evaluate` member function, which takes an integer parameter and returns the value of the polynomial represented by the `PolynomialLinkedList` at the parameterized value.
- A `derivative` member function, which returns the `PolynomialLinkedList` corresponding to the polynomial's first derivative (e.g., for polynomial $3x^5 - 12x^3 + x - 19$, the derivative is $15x^4 - 36x^2 + 1$).
- Overloaded operators for adding (+) and multiplying (*) two `PolynomialLinkedLists`, as well as for standard input (<>) and output (>>) to streaming files, where the I/O uses separate lines for each monomial’s pair of integer values.
- A `fullOutput` member function that outputs a formatted version of the polynomial, with the caret symbol (^) used to denote exponentiation in the quadratic and higher exponents. Note that unit coefficients should not be displayed and that the leading coefficient should not be preceded by a plus sign.

You will also implement a driver program in order to test this class. An example of satisfactory output is displayed on back of this page.

Take note of the following:

- Unlike the word-count example presented in class, you do not need to use recursion on this program.
- Familiarity with calculus is not a prerequisite to this class, so if you are unfamiliar with taking derivatives of polynomials, consult the instructor for assistance.
- To simplify the addition and multiplication operators between pairs of `PolynomialLinkedLists`, it is strongly suggested that you also write addition and multiplication operators for a `PolynomialLinkedList` and the monomial structure. Adding two polynomials can then be implemented as a sequence of monomial additions, while multiplying two polynomials can be implemented as a sequence of sums of polynomial-monomial products.
- Code comments are required in your program. They fall into three distinct categories:
  - Each program file must begin with a comment containing your name, the assignment number, and a paragraph (3-4 sentences) explaining the contents of the program file.
  - Each function must be preceded with a sentence or two, explaining the purpose of the function.
  - Any code segment that is particularly difficult to decipher upon reading for the first time should be preceded with an explanatory comment that clarifies its meaning.
- Standard readability and modularity practices are required in your program. Skipped lines, indentation, and meaningful variable names can improve readability. Redundant code, lengthy functions, and excessive cut-and-paste tend to damage a program’s modularity.
Zip-compress your entire program folder and place it on your dropbox by the deadline mentioned above. Several files of test data are available on the course website.

You must write your own code on this assignment, with adequate explanatory documentation. Obtaining code assistance from any outside source is considered academic misconduct. The only person permitted to see your code prior to the assignment deadline is the instructor.