CS 255 – Fall 2021 - University of North Alabama

Project 3: Build Your Own Class – Polynomial, as a Linked List

Source Code Due Date: Tuesday, November 30, 2021 at 11:59 pm.

10-point Bonus Point Due Date: Monday, November 22, 2021 at NOON (+ 10 points on project)

- **Upon early submission, make a comment "10-point Bonus" on the assignment comments**

5-point Bonus Due Date: Tuesday, November 23, 2021 at 11:59 pm.

- **Upon early submission, make a comment "5-point Bonus" on the assignment comments**

Late Policy: You may submit a project late for up to 24 hours after the deadline with an automatic 20% reduction from the total possible points. You will have a 30-minute grace period to account for any time discrepancies. No work will be accepted after the late date.

Submission: Upload file (poly.cpp) to Canvas, for this project.

Compiler: I will compile your project in Dev C++ using my Driver, assuming Poly.h is implemented in your Poly.cpp.

Problem: You will create the implementation of a Polynomial data type using a Linked List.

Specifics: I will provide the Polynomial class specification (class declaration), the Polynomial class Implementation shell (class definition), a driver program to test your Polynomial class, and input files. You will complete the Poly methods. You may draw from the complete LinkedLink implementation from class. If you use direct code, make a comment that you pulled specific pieces of code from that source.

LinkedList and Node data representation:

Your Polynomial will be represented by a linked list. Each term in a polynomial will be represented by a Node object. This means your Node data section will need to be modified as each term will have a decimal coefficient and an integer coefficient.

Node Operations Upon the Data:

Constructors only
Make the Linked List (Polynomial) class a friend of the Node class
Make all operations private

Linked List Operations Upon the Data:

1. Constructor: head to NULL
2. AddTerm: You should have a routine that takes in a coefficient, and an exponent, creates a Node for that term and inserts the Node into the list in the appropriate place. The list should be maintained in descending order with respect to the exponent (the highest exponent at the head of the list). You should not implement the InsertAtFront, InsertAtEnd, InsertAtPos methods to this class. Duplicate exponents (degrees) are allowed. In this case, the coefficient should be added to the existing coefficient.

Linked List Operations Upon the Data, cont'd:

3. Copy Method: This will be a method used to help make a deep copy. Implement the deep copy code once and then you may use it in the Copy Constructor and the Assignment Operator.

4. Copy constructor: Make a deep copy of the object rather than a shallow copy.
5. Assignment Operator: Appropriately handle the old copy then make a new deep copy.
6. Destructor: Deallocate the Poly and set head to null.

7. Input Operator: Input Polynomials in the Following format
   <Cx^D [+Cx^D]>
   Example: The Polynomial listed above would be

   <1x^20 + 3x^16 + 15x^4 + 2x^3 + 16x^2 + 5x^0>

   You may assume all Polys will be in this format.

8. Output Operator: Output Polynomials in the same format as the input.
   You may should print the term with degree 0 without the x^0.

9. Reset: All Nodes should be deallocated and head reset to NULL. This method can be used in the Assignment operator and in the Destructor

10. Evaluate: Given x, evaluate the polynomial.

11. Addition operator: Return the sum of the two given polynomials. Hint: Check you Add Routine

12. Derivative: Take the derivative of the calling object. The data will be changed after this method is called.

Comments:

Comment Header: You should have a header for each file that contains your name, the name of the file, the description of the file (for this project, you will include the project description), the course number, and the due date.

Throughout: Comment major segments of code. Comment any identifiers that do not clearly identify a variable or constant. Comment formulas. Comment any outside sources.

Functions: Use the format listed in the design plan.

Other notes:

Do not use global variables.

Keep data members private.

Do not change the Poly class without prior permission.

As applicable in any project, you should design your code in such a way that it is reusable. Make appropriate use of functions and reuse existing code where possible for your driver.