Project.pdf

Directions: You must upload a single zip file (e.g., Liu-Proj.zip) to blackboard. The file should contain the following:

a. All *.cpp and *h files
b. Output/result
c. A readme file that contains the philosophy/concepts/descriptions of how you implement this project, and how to run your project. (IF YOU DON'T HAVE THE README FILE, YOU WILL GET 10 PTS OFF). You may also simply write as many comments as possible in your code.

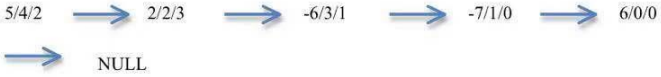
Please DO NOT submit the entire Visual Studio projects to me.

Honor Statement: By turning in your assignment to me, you confirm that all of the work is **your own**. That is, you did not take your answer and adapt it from someone else, or get detailed advice from another student on how they answered a question. You also cannot give away your answers to another student. You agree, by sending me the answers via email, that this is your own work. This also includes trying to find an answer on the web (if such answers exist, I will find all of them and make sure that your answer is unique). **Penalties for cheating and plagiarism range from a 0 or an F on a particular assignment, through an F for the course, to expulsion from the university.**

Linked List (Singly, double, or circular) can be used to represent a polynomial expression, in a single variable x. for example,

f = 5x^4*y^2 + 2x^2*y^3 - 6x^3*y^1– 7x + 6

can be expressed as:



NULL

1. Introduce a Node class and a LinkedList class that can be used to express the above expression.
2. Introduce a function, called *degree*, in LinkedList class that will return the highest power of a term with a nonzero coefficient (e.g., degree() will return 6 (4+2) based on the above expression).
3. Introduce a function, called *coefficient*, in LinkedList class that will return the coefficient of i-th term (e.g., coefficient(0) will return 5. We assume that an expression should be ordered from high to low degree as shown in the diagram).
4. Introduce a function, called *match*, in LinkedList class that will return true if the two expressions are the same. Otherwise, return false. For example,
   y = x^4*y^2 – 6x^3*y^1 + 2x^2 – 7x + 6
   z = x^4*y^2 – 6x^3*y^1 + 2x^2 – 7x + 6
   a = x^4*y^2 – 6x^3*y^1 + 2x^2 – 7x

   y.match(z) will return true
   y.match(a) will return false

5. Introduce a function, called *sum,* in LinkedList class that will update the current LinkedList object. For example,

   y = 5x^4*y^2 – 6x^3 + 2x^2*y^1 – 7x + 6
   z = 3x^2*y^1 + 4x*y^2 + 7
   y = y.sum(z) will update y as
   5x^4*y^2 – 6x^3 + 5x^2*y^1 + 4x*y^2 - 7x + 13
6. Introduce a function that converts an infix expression of Linked List into postfix expression of Linked List.
7. Introduce a function, called *dot,* in LinkedList class that will update the current LinkedList object. For example,

   y = 5x^4*y^2 – 6x^3 + 2x^2*y^1 – 7x + 6
   z = 3x^2*y^1 + 4x*y^2 + 7
   y = y.dot(z) will be compute as follows
   (5x^4*y^2 – 6x^3 + 2x^2*y^1 – 7x + 6) * (3x^2*y^1 + 4x*y^2 + 7)
   = 15x^6*y^3 + 20x^5*y^4 + 35x^4*y^2 – 18x^5*y^1 – 24x^4*y^2 – 42x^3 + 6x^4*y^2 + 8x^3*y^3 + 14x^2*y^1 – 21x^3*y^1
   -28x^2*y^2 – 49x^1 + 18x^2*y^1 + 24x^1*y^2 + 42……..(1)
   = 15x^6*y^3 + 20x^5*y^4 + 17x^4*y^2 – 18x^5*y^1 + 8x^3*y^3 – 21x^3y^1
   -28x^2*y^2 - 42x^3 +32x^2*y^1 + 24x*y^2 - 49x + 42. …….(2)

   Please note once computation is done as seen in (1) you need to simplify expression to (2). Otherwise, you will get points off.