Team 1 Project 1B

### PREPARED BY

Joe Simon

Eric Vaughn

James Demeritt

Jeff Kimani

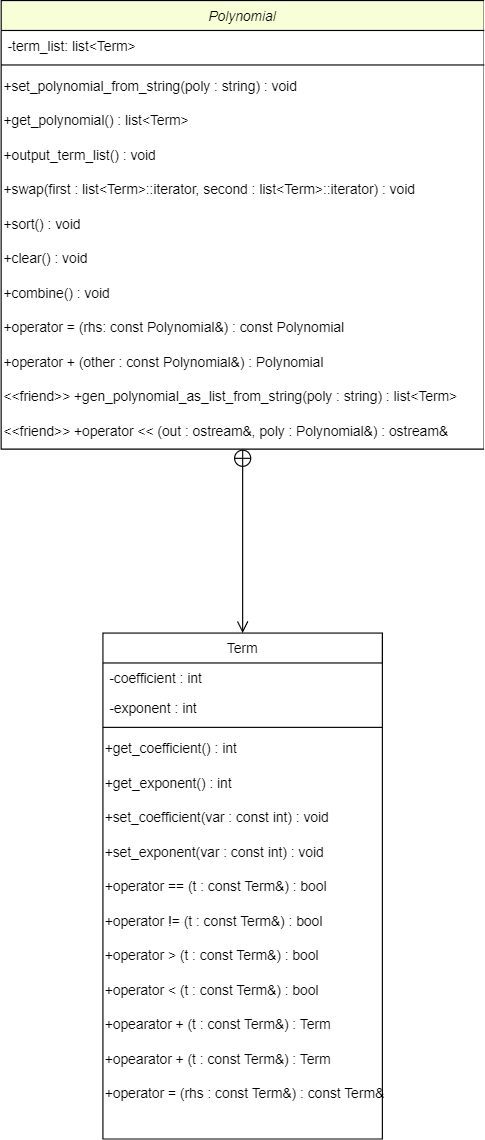
2) System Design

The system is designed by creating a class Polynomial with a child class Term, which has two variables (int coefficient and int exponent). The Polynomial class has a term\_list variable, which has a type of list<Term>. All functions inside the Polynomial class and Term child class are declared in the “Polynomial.h” file, and are initialized inside of the “Polynomial.cpp” file.

Inside the “Main.cpp” file, three Polynomial objects are created. The first two Polynomial objects take in a string, first\_p and second\_p, which are then used to instantiate the term\_list variable of both the first and second Polynomial objects through the set\_polynomial\_from\_string() function. After this, the third Polynomial object is instantiated through the overloaded “+” operator, which adds two Polynomial objects, which can be shown as “first + second;”. Once the first and second Polynomial object have been added together, the third polynomial storing the result will be sorted through the .sort() function, and then will be combined with the .combine() function. After this, the third polynomial is output into console using the overloaded “<<” operator.

Two data structures were implemented in the system; the doubly-linked list data structure and the iterator data structure. The doubly-linked list data structure is used inside the polynomial class to store all Terms of the polynomial. Once all of the Terms have been stored inside of term\_list, then the iterator data structure is used to iterate through the doubly-linked list to grab each Term inside of term\_list. This allows us to access the coefficient and the exponent of each Term.

3) UML Class Diagram



4) Two Test Cases

Each test case can be found inside the GitHub repository.

# Test Case 1:

This test case is going to test the Polynomial class's .sort() function.

for s\_1, we would expect it to become sorted from:

"-x+5+x^2-10"

to

"x^2-x+5-10"

The constant values (5 and -10) might swap places, since we are sorting exclusivly by the term's exponents.

I will run the program and see what we get...

We got:

"x^2-x-10+5"

Nice! As expected, they are ordered by their term's exponents. The constants are swapped but that isn't a surprise (or a problem).

--------------------

Let's try s\_2.

The expected output would be:

"5x-5x^-2+10-5x+x^2-0x"

to

"x^2+0x-5x+5x+10-5x^-2"

Let's run it...

We got:

"x^2+0x-5x+5x+10-5x^-2"

Wonderful! The -0x changed into a 0x, but there is no such thing as negative 0, and adding or subtracting by 0 has the same result.

# Test Case 2:

This test case is going to test the overloaded '+' operator.

for the first polynomial, we expect an output of:

"First Polynomial: -3x^7+2x^7-83x^12+30"

and we got:

"First Polynomial: -3x^7+2x^7-83x^12+30"

The actual output is correct!

Next, we're going to test the second polynomial. We should get an output of:

"Second Polynomial: 45x^2-8x+0x-0x^-3"

Let's see the actual result:

"Second Polynomial: 45x^2-8x+0x-0x^-3"

The actual result is good! Now for the real test; Adding them both together, sorting, and combining. We should get the following result:

"Printing result: -83x^12-x^7+45x^2-8x+30"

Notice how it should get rid of the "+0x-0x^-91". Let's see what we actually get.

"Printing result: -83x^12-1x^7+45x^2-8x^1+30"

Looking at the actual result and comparing with the expected result, we can see a few key differences.

The "-x^7" is now "-1x^7", but those are the same thing, so nothing to worry about. Another difference is that "-8x" is now "-8x^1".

As with the previous term, this isn't that big of a deal as they are both the same thing.

5) Contributions

1. Joe
   1. Helped create the Main.cpp file and created the “test\_case\_2.cpp” file, declared set\_polynomial\_from\_string(), output\_term\_list(), sort(), combine(), operator +(), and operator <<() functions. Initialized set\_polynomial\_from\_string(), output\_term\_list(), operator +(), and operator <<() functions. Helped write Project report. Created most of the UML Diagram.
2. Eric
   1. Created “Polynomial.h”, “Polynomial.cpp”, and “test\_case\_1.cpp” files. Created the Constructor, Copy Constructor, and Destructor for the Polynomial class and the Term class. Declared the Polynomial and Term classes inside “Polynomial.h” file, Initialized sort(), clear(), combine(), and get\_polynomial() functions in the Polynomial public section. Redefined the set\_polynomial\_from\_string() and operator +() functions. Declared and initialized the variables for the Term class and get\_coefficent(), get\_exponent(), set\_coefficent(), and set\_exponent(), operator ==(), operator !=(), operator >(), operator <(), operator +(), and operator =() functions. Helped write the Project report. Helped finish the Main.cpp file. Helped finish the UML Diagram.
3. James
4. Jeff

6) Improvements

1. The code came out to be ~500 lines long in the “Polynomial.cpp” file, which I feel could definitely be reduced. As a result, the total size of the project came out to be 753MB, so if space was a concern, then this is what we would have to focus on.
2. As shown in test\_case\_2, we could make the overloaded << operator show “-x^7” instead of “-1x^7”. Same goes with “-8x” and “-8x^1”, even though they mean the same thing.
3. The .sort() function for the Polynomial class could be improved. Currently, we are using the std::list<T> .sort() and .reverse() functions for its implementation. We need the .reverse() function because the standard .sort() sorts by least to greatest (Term-wise) and we needed it to be greatest to least. We would need to write our own sorting algorithm for this so we wouldn’t have to reverse the sorted list.
4. The Main.cpp file could be extended to allow the user to add Polynomials until they wanted to quit the program.
5. We could add an output file that the user can save their Polynomial sums to, for later use outside of our program.