Programming Assignment 1

Polynomials

Posted Fri, Sep 16

Due Fri, Sep 30, 11:00 PM (WARNING!! NO GRACE PERIOD).

Extended deadline (with ONE time free extension pass): Mon, Oct 3, 11:00 PM (NO GRACE PERIOD)

Worth 60 points (6% of course grade)

In this assignment, you will implement polynomials and operations on them using a linked list.

- You will work **individually** on this assignment. Read the DCS Academic Integrity Policy for Programming Assignments - you are responsible for this. In particular, note that **"All Violations of the Academic Integrity Policy will be reported by the instructor to the appropriate Dean".**
- You get ONE extension pass for the semester, no questions asked. There will be a total of 5 assignments this semester, and you may use this one time pass for any assignment EXCEPT the last. **A separate Sakai assignment will be opened for extensions AFTER the deadline for the regular submission has passed.**
- **IMPORTANT - READ THE FOLLOWING CAREFULLY!!!**

  Assignments emailed to the instructor or TAs will be ignored--they will NOT be accepted for grading. We will only grade submissions in Sakai.

  If your program does not compile, you will not get any credit.

  Most compilation errors occur for two reasons:

  1. You are programming outside Eclipse, and you delete the "package" statement at the top of the file. If you do this, you are changing the program structure, and it will not compile when we test it.
  2. You make some last minute changes, and submit without compiling.

  To avoid these issues, (a) START EARLY, and give yourself plenty of time to work through the assignment, and (b) Submit a version well before the deadline so there is at least something in Sakai for us to grade. And you can keep submitting later versions (up to 10) - we will accept the LATEST version.

Background

Read Section 3.1 in the textbook for background on polynomials and polynomial arithmetic.

A polynomial may be represented using a linked list as follows: for every term in the polynomial there is one entry in the linked list consisting of the term's coefficient and degree. The entries are ordered according to ASCENDING

values of degree, i.e. lowest degree term first, then next lowest degree term and so on, all the way up to the highest degree term. IMPORTANT: Zero-coefficient terms are NOT stored.

For example, the following polynomial (the symbol '^' is used to mean 'raised to the power'):

4x^5 – 2x^3 + 2x +3

can be represented as the linked list of terms:

(3,0) -> (2,1) -> (-2,3) -> (4,5)

where each term is a (coefficient,degree) pair.

Notes about representation:

- Terms are stored in ASCENDING order of degrees from front to rear in the a non-circular linked list.
- Zero-coefficient terms are NOT stored.
- An EMPTY (zero) polynomial is represented by a linked list with NO NODES in it, i.e. referenced by NULL.
- Coefficients are floating point numbers
- Degrees are POSITIVE integers, except if there is a constant term, in which case the degree is zero.
- There will not be more than one term in the same degree.

If you do not represent all your polynomials (the initial inputs as well as those you get out of doing arithmetic on polynomials) as above, you will lose credit even if your results are mathematically correct.

Implementation and Grading

Download the attached polynomial_project.zip file to your computer. DO NOT unzip it. Instead, follow the instructions on the Eclipse page under the section "Importing a Zipped Project into Eclipse" to get the entire project into your Eclipse workspace.

You will see a project called Polynomial with the following classes in package poly:

- Polynomial
- Polytest

(Aside from these, there are also three sample input files, described in the **Running the Program** section below.)

The file Polynomial.java contains two other classes:

- A class called Term that is used to implement each term of a polynomial, with fields for coefficient and degree
- A class Node that implements a linked list node, and contains a Term object.

Using these two classes, you need to fill in the implementation of the Polynomial class where indicated in the Polynomial.java source file. This includes the following:

Method	Grading Points
evaluate	10
add	25
multiply	25

Note: You will NOT get any credit if you convert the polynomial representation to an array, work on arrays, then convert back to linked lists. You must work with linked lists ONLY all the way through.

Observe the following rules while working on Polynomial.java:

- Only fill in the code in the methods add, multiply, and evaluate where indicated.

- In methods that return a Polynomial (add and multiply), the polynomial that is returned must be represented as described in the "Notes about representation" part of the **Background** section above.
  Your method will not get credit if the returned polynomial does not adhere to this representation, even it is mathematically correct.
  Also see the "Notes about empty (zero) polynomials" at the end of the **Running the program** section below.
- **DO NOT** remove the import statements at the top of any of the given classes.
- **DO NOT** change the headers of ANY of the given methods
- **DO NOT** change/remove any of the given class fields
- **DO NOT** add any new class fields.
- **DO NOT** import any packages other than those already imported in Polynomial.java
- **YOU MAY** add new helper methods, but you must declare them **private**.

Running the program

Here are three sample input files for you to test (they should be under the project folder in Eclipse):

- A file ptest1.txt that contains the polynomial

  4x^5 - 2x^3 + 2x + 3

- A file ptest2.txt that contains the polynomial

  8x^4 + 4x^3 - 3x + 9

- A file ptest1opp.txt that contains the polynomial

  -4x^5 + 2x^3 - 2x - 3

  (the negation of the polynomial in ptest1)

In each of these files, each line is a term, with the first value being the coefficient, and the second value being the degree. The terms are listed in **descending** order of degrees and the respective non-zero coefficients. Remember that when you store a polynomial in a linked list, you will store it in **ascending** order of degrees. (This is actually already implemented by the Polynomial constructor when it reads a polynomial from an input file. All you have to do is make sure you stick with this rule when you add and multiply.)

You may assume that we will NOT test with an invalid polynomial file, i.e. every test input file will either have at least one term in it in the correct format, or will be empty (see **Notes about empty (zero) polynomials** below). So you don't need to check for validity of input.

Here's a sample run of the driver, Polytest. Apart from ptest1.txt, ptest2.txt, and ptest1opp.txt, a fourth test polynomial file, ptestnull.txt is also used. This is an empty file that stands for a null (zero) polynomial. See notes after the test run for special instructions regarding zero polynomials.

```
Enter the name of the polynomial file => ptest1.txt

4.0x^5 + -2.0x^3 + 2.0x + 3.0


1. ADD polynomial
2. MULTIPLY polynomial
3. EVALUATE polynomial
4. QUIT
    Enter choice # => 1
Enter the file containing the polynomial to add => ptest2.txt

8.0x^4 + 4.0x^3 + -3.0x + 9.0
```

```
Sum: 4.0x^5 + 8.0x^4 + 2.0x^3 + -1.0x + 12.0


1. ADD polynomial
2. MULTIPLY polynomial
3. EVALUATE polynomial
4. QUIT
     Enter choice # => 1
Enter the file containing the polynomial to add  => ptest1opp.txt

-4.0x^5 + 2.0x^3 + -2.0x + -3.0

Sum: 0


1. ADD polynomial
2. MULTIPLY polynomial
3. EVALUATE polynomial
4. QUIT
     Enter choice # => 1
Enter the file containing the polynomial to add  => ptestnull.txt

0

Sum: 4.0x^5 + -2.0x^3 + 2.0x + 3.0


1. ADD polynomial
2. MULTIPLY polynomial
3. EVALUATE polynomial
4. QUIT
     Enter choice # => 2
Enter the file containing the polynomial to multiply  => ptest2

8.0x^4 + 4.0x^3 + -3.0x + 9.0

Product: 32.0x^9 + 16.0x^8 + -16.0x^7 + -20.0x^6 + 52.0x^5 + 38.0x^4 + -6.0x^3 + -6.0x^2 + 9.0x + 27.0


1. ADD polynomial
2. MULTIPLY polynomial
3. EVALUATE polynomial
4. QUIT
     Enter choice # => 3
Enter the evaluation point x  => 2
Value at 2.0: 119.0


1. ADD polynomial
2. MULTIPLY polynomial
3. EVALUATE polynomial
4. QUIT
     Enter choice # => 4
```

The sample tests we have given you are just for starters. You will need to create other tests of your own on which you can run your code. For every test you run, be careful to keep your test input in the same format as the test files provided, otherwise the driver will not work correctly. And make sure your test file is in the same folder as the other files, i.e. under Polynomial.

Note on translation from internal to output representation:

The toString method in the Polynomial class returns a string with the terms in descending order, fit for printing. So, you don't need to write a separate nethod to do this. For illustration, see how the add method in Polytest prints the resulting polynomial:

```
System.out.println("Sum: " + p1.add(p2) + "\n");
```

p1.add(p2) returns the result of adding p2 with p1, and because this result polynomial is placed in a context that expects a string, the toString method is called on it, which returns a reversed representation of the linked list.

Notes about empty (zero) polynomials:

- If you want to test with an empty polynomial input, you should create a file with nothing in it. In Eclipse, you can do this by right clicking on the project name in the package explorer view, then selecting **New**, then selecting **File**. Give a name, and click **Finish**. You new file will show up under the project name folder in the package explorer view, and the file will be opened in the text editor view. But don't type anything in the file.
- Remember that when you add two terms of the same degree, if you get a zero coefficient result term, it should not be added to the result polynomial. As listed in the "Notes about representation" in the **Background** section, zero-coefficient terms are not stored.
- The string representation of a zero polynomial is "0" - see the toString method of the Polynomial class. So, the Polytest driver will print a zero for a zero polyomial input, or a zero polynomial that results from an operation performed on two polynomials.

Submission

Submit your **Polynomial.java** source file (NOT Polynomial.class), in Sakai -> Assignments.

Refer to the instructions in the **Eclipse** page, under the section **The Eclipse Workspace** to know how to locate **Polynomial.java** on your computer for uploading.

Grading Process

Your submission will be auto-graded by a grading script that will run several test cases on each graded method. For each test case, the result computed by your code will be compared with that computed by our correct code.

Note that for the add and multiply the comparison with the correct result is based on the state of the polynomial linked list in the program, NOT on any printed output. In other words, the grading script will compare the linked list structure of the correct result with the linked list structure in your implementation.

All printed output will be ignored. This also means if you threw in print statements for debugging and left them in your code, they will have no bearing on the grading.

When grading is done, your test report will be emailed, detailing the score on each test case. Test cases will be posted so you can run your program against them to verify the test report. Remember, verification means checking the state of the polynomial linked list for add and multiply, NOT what your program might print.