Project #2

CIS 2353 - Prof. John P. Baugh – Oakland Community College – OR
Fall 2018

Points: ____________ / 125
Due: November 6, 2018 at 11:59 p.m.

Part I: Programming

Objectives

- To work with linked chains of nodes
- To create a component of mathematics solving software

Instructions

For this assignment, you will create a class, named Polynomial, and a class named Node. Each Polynomial object will be responsible for maintaining a chain of linked nodes. You can assume the polynomial is in standard, decreasing term form. For example, a polynomial will always be read from file in the correct order, and constructed as for example, $4x^2 + 2x + 3$ and **not** something like $2x + 3 + 4x^2$. Each term (monomial) will be represented by a single Node object. For example, $4x^2$ would be one Node object, $2x$ is another, and 3 is another.

Assume all polynomials represent a function of x, and no other mathematical variables. In other words, x will be the only character encountered representing a mathematical variable.

Node class

The Node class represents an individual term of a polynomial. You must at least contain data representing the Polynomial, and a link to the next Node object in the chain. The chain should be singly-linked, with each node pointing to the next node.

To keep things simple, we assume all powers and coefficients are integers.

The fields you must maintain will consist of:

- exponent, an integer
- coefficient, an integer
- nextNode, a reference to the next Node object

Polynomial class

Method	**Notes**
Polynomial()	No-argument constructor. Sets the polynomial's linked chain head node to null
Polynomial(String poly)	The poly argument is a string representing a polynomial, in standard, decreasing term form, such as 4x^2+2x+3. The constructor is ultimately responsible for creating the individual nodes and chaining them appropriately.
Polynomial(Polynomial otherPoly)	A copy constructor. This constructor will perform a **deep copy** on the otherPoly, making an exact copy of it.
print()	The print method simply prints out the polynomial represented by the object. Given a Polynomial object, myPoly, the call to myPoly will simply print out the polynomial by walking down the linked chain. Assuming all printing is done to the console.
add(Polynomial poly1, Polynomial poly2)	This *static* method takes two Polynomial objects as arguments and returns a new Polynomial object, that is the sum of the other two. Remember that like-terms must be combined. Also, note that not all polynomial will be of the same degree.

File and User Input

The polynomials will be constructed from one file, **polynomials.txt.** Each Polynomial object may be placed in an ArrayList or other data structure. The contents of all Polynomials should be listed to the screen, each on their own line immediately after they are done being constructed.

For example, if the file contains:

```
4x^2+2x+3
5x^3+15
2x^2+2x+5
```

The following is printed out to the screen based on an iteration through the ArrayList:

```
List of Polynomials:
0:  4x^2+2x+3
1:  5x^3+15
2:  2x^2+2x+5

Which do you wish to add?  Press -1 to Exit.
```

The numbers in front of the polynomials are just the indices in the ArrayList at which each polynomial resides.

The prompt asks the user which ones they wish to add, or if they want to exit. -1 causes an exit. Any other input will be in the form: *int1 int2*, where each is an integer.

If the user enters an invalid non-zero positive number (i.e., an index out of range), just print that the input was invalid, and prompt them again.

If the user enters two integers indicating the polynomials to add, your program should add them, creating another Polynomial object, and will automatically add it to the ArrayList, and print out all the polynomials again, prompting the user once again for their input.

For example, BEFORE the addition:

```
List of Polynomials:
0:  4x^2+2x+3
1:  5x^3+15
2:  2x^2+2x+5

Which do you wish to add?  Press -1 to Exit.
0 2
```

This will cause polynomials at 0 and 2 to be added, and the screen should display the following:

```
List of Polynomials:
0:  4x^2+2x+3
1:  5x^3+15
2:  2x^2+2x+5

Which do you wish to add?  Press -1 to Exit.
0 2

List of Polynomials:
0:  4x^2+2x+3
1:  5x^3+15
2:  2x^2+2x+5
3:  6x^2+4x+8
Which do you wish to add?  Press -1 to Exit.
```

This will continue until the user exists with a -1. (**Technically, you can make any negative number cause an exit, but you don't need to tell the user that.)**

Part II: Analysis Questions

For this, you will solve problems related to efficiency, and the Big O asymptotic notation. **SHOW YOUR WORK.**

The definition of Big O

Let f(n) and g(n) be functions mapping positive integers to positive real numbers.
We say that f(n) is O(g(n)) if there is a real constant $c > 0$ and an integer constant $N \geq 1$ such that:

$$\mathbf{f(n) \leq c * g(n), \quad for\ n \geq N}$$

Note that the constants c and n_0 are *not unique*. You just have to find a *c* and an *N* that satisfies the definition of Big O

3.1 Given a time function, $T(n) = 3x^2 + 5x + 2$, find constants **c** and **N** that prove that the big O of the growth function T(n) is n^2

3.2 Find the Big O of $T(n) = 4x^3 + 12x^2 + 2x + 12$. Justify your answer by finding constants c and N

3.3 Find the Big O of the following code:

```
for(int i = 0; i < n; i++)
{
    for(int j = 0; j < n; j++)
    {
        System.out.println("Hello!");
    }
}
```

You do not have to find a T(n), or c and N. Just give the Big O, and informally justify your answer.

Deliverables

Turn in a **zipped up folder**, containing the Java file(s) and any required additional files required for this project. This includes all relevant classes that are created, as well as the client used to test the class(es) – i.e., the class containing the main method. Upload them to D2L under Assignments, to the appropriate assignment directory.

Make sure your name and course information are at the top of each Java file. For example:

```
//  Tina Tarantula
//  CIS 2353
//  Fall 2018
//  Project 2
```