**CS 248 – Object-Oriented Programming and Data Structures**

**HW4: 100 points**

**Objective:**

The objective of this homework is to learn about:

1. Develop knowledge regarding different operations of link list
2. Identify the run time efficiency of algorithms

**Problem Statement:**

A polynomial is of the form: 

Where, ci is the coefficient of the ith term and n is the degree of the polynomial. Some examples are:



It is not necessary to write terms of the polynomials in decreasing order of degree. But for this assignment, we will input them in this decreasing order. The computer implementation requires implementing polynomials as a list of pairs of coefficients and exponent. Each of these pairs will constitute a node structure, so a polynomial will be represented as a list of nodes. A linked list structure that represents polynomials 5x4 – 8x3 + 2x2 + 4x1 + 9x0 illustrates in the following figure:



**Addition of Polynomials:**

To add two polynomials, we need to scan them once.

1. If we find terms with the same exponent in the two polynomials, then we add the coefficients and print the result. Go to the next node for both the polynomials.
2. Otherwise we print the term of larger exponent and go to the next node of the link list that contained this larger exponent.
3. When we reach at the end of one of the polynomials, then remaining part of the other is printed.

To add two polynomials, follow the following steps:

• Read two polynomials

• Add them

• Display the resultant polynomial

**Task 1:**   Write the following functions:

**public void insert\_at\_end(int c, int e) :** Here the first parameter is the coefficient, and the second parameter is the exponent.

**public void traverse();**

You also need to modify the node class. Up to this part, you can simply modify the SingleLinklist code we did before.

**public static void add\_poly(Poly x, Poly y)**: This function takes two Poly type objects as input parameters and does the actual polynomial addition. Also, it prints the final result. (Please note it is a static method, so it can be called using the class name)

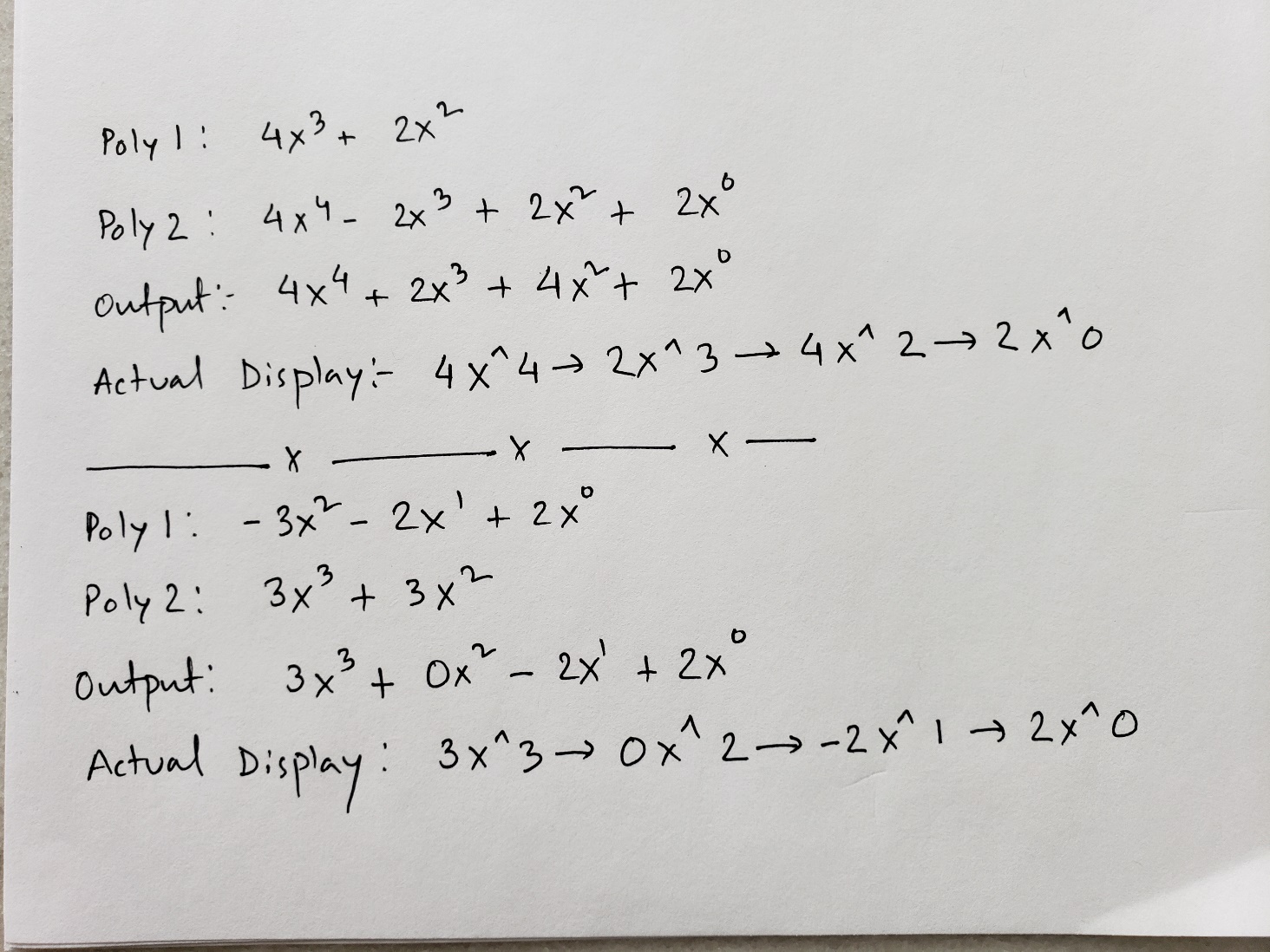
All the required functions and their input and output are described in the starter code.

**Task 2:**   What is the Big O of the function **public static void add\_poly(Poly x, Poly y)**? Explain. (assume, x has m elements and y has n elements)

**Input Format:** You need to read from a file named **poly.txt**. First line of the file contains the number of nodes of the first polynomial - *N*. Second line contains *2N* integers (separated by space), which represent the coefficient and exponent of each node of the first polynomial. Third line of the file contains the number of nodes of the second polynomial - *M*. Fourth line contains *2M* integers (separated by space), which represent the coefficient and exponent of each node of the second polynomial.

**Output Format:** Output should be printed in the console.

**Sample input output:**



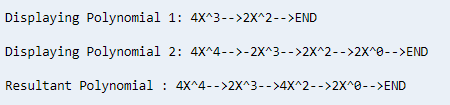
**Sample input/output when you run the program:**

2

4 3 2 2

4

4 4 -2 3 2 2 2 0



**HW Grading:**

1. 5% - Follows style guidelines, including header and correct filename.
2. 40% - Compiles without errors
3. 45% - Correct implementation of all functions
4. 10% - Report shows explanation of the order

**Bonus (10 points – capped at 100):** Rewrite the **public static void add\_poly(Poly x, Poly y)** function as an instance method (as shown in Time class example) and call it from the PolyMain function. It should print the same result.