CS 2413 – Data Structures – Spring 2019 – Project Three
Due: 11:59 PM, March 13th, 2019

Objective: The main objective of this programming project is to create and manipulate linked list class.

Description: There are two parts to this project:

1. **Create a Linked List class (that is templated) that allows the following operations.**
   a) empty constructor and copy constructor
   b) destructor
   c) overloaded = operator and the [] operator, where [i] returns the ith element of the linked list
   d) insert method
   e) remove method – given the index position remove the object in that position.
   f) size method – number of items that are in the linked list
2. **Implement the polynomial class described below.**

This part of the project is about manipulating polynomials on a single variable. You will be required to construct a polynomial class that will use the linked list that you have built. You will keep the terms in the polynomial sorted in the descending order of the exponent of the terms. You will write methods to implement operations such a **multiplication and addition** of two polynomials, evaluation of a polynomial given the value for the single variable, insertion and deletion of terms in a polynomial and printing the terms in the polynomial.

A polynomial is sum of terms, where each term is of the form cx^e, where c is the coefficient, x is a variable, and e is the exponent. The largest exponent in a polynomial is called the *degree* of the polynomial. For example, $A(x) = 2x^6 + 10x^3 - 2x + 4$ is a polynomial containing four terms with non-zero coefficients and the degree of the polynomial is 6. This polynomial can be represented using a linked list A of size 4. The array A for the polynomial A(x) given above looks as below.

0	1	2	3
(2,6)	(10,3)	(-2,1)	(4,0)

The addition of two polynomial $A(x) = \sum a_i x^i$ and $B(x) = \sum b_i x^i$ is defined as $A(x) + B(x) = \sum (a_i + b_i)x^i$.
The multiplication of the polynomials A(x) and B(x) is defined as $A(x).B(x) = \sum(a_i x^i) \times \sum (b_j x^j))$.

Before you create a polynomial class, you will create a term class that all the standard method that any class should have:

```
class Term {
      protected:
            int coefficient;
            int exponent;
      public:
            /* all the necessary methods */
};

class Polynomial {
      protected:
            LinkedList<Term>* myPoly;
      public:
            /* all the necessary class methods and the methods you are
            asked to implement */
};
```

Write methods as part of the polynomial class to implement the following operations.

- evaluatePoly (x) – Given a value for x evaluate the polynomial and print the result.
- addTerm (coefficient, exponent) – add a term to the polynomial and make sure that the polynomial is kept in sorted order of the exponent.
- deleteTerm (exponent) – delete the term, that is set the coefficient to be zero for the term containing the exponent specified as the parameter.
- addPolynomial (polynomial) – Perform polynomial addition with the polynomial specified in the parameter and create a new polynomial.
- You are to overload the + operator
- multiplyPolynomial (polynomial) – Perform polynomial multiplication with the polynomial specified in the parameter and create a new polynomial.
- You are to overload the * operator.
- printPolynomial () – Print the polynomial in a suitable form such that polynomial can be read.
- Overload the << operator.

Input: At any given time the program will work with only two polynomials. The input for your program consists of an unknown number of lines of input data. Each line of the input data contains a character symbol and followed by appropriate data. Each character in the input represents either a command A (for adding the two polynomials), M (for multiplying the two polynomials), E (evaluate the polynomial given a value for x), D (delete a term in a specified polynomial) I (insert a term in a specified polynomial) or P (print the specified polynomial in sorted order of the exponent). Sample input follows:

I 1 –2 1	→ insert a term with coefficient –2 and exponent 1 in polynomial 1.
I 1 4 4	→ insert a term with coefficient 4 and exponent 4 in polynomial 1.
I 2 8 11	→ insert a term with coefficient 8 and exponent 11 in polynomial 2.
I 2 3 1	→ insert a term with coefficient 3 and exponent 1 in polynomial 2.
I 1 4 1	→ term with exponent 1 exists in polynomial 1, add 4 to the existing coefficient.
P 1	→ print the first polynomial
P 2	
A 1 2	→ add the first and the second polynomial
I 1 3 3	
A 1 2	
M 1 2	→ multiply the first and the second polynomial
E 1 10	→ evaluate the first polynomial with the variable value as 10
E 2 5	
D 2 3 0	→ there is no term in polynomial 2 with an exponent of 0 (print this message)
D 2 3 1	
P 1	
M 1 2	

Processing: After you read in a line of input, process the input according to the specified input command. Your main function may appear (note that complete details are not given) as below.

```
// Create an array of polynomials the maximum being 10.
char command;
int polynum, coefficient, exponent; value;

cin >> command;
while (!cin.eof())
{
  switch (command)
  {
        case 'I': cin >> polynum >> coefficient >> exponent;
                  Polynomials[I].addTerm (coefficient, exponent);
                  break;
        case 'D': ….; break;
        case 'A': cin >> i >> j; Polynomials[i] + Polynomials[j]; break;
        case 'M': cin >> i >> j; Polynomials[i] * Polynomials[j]; break;
        case 'E': cin >> polynum >> value;
                   Polynomials[polynum].evaluatePoly(value);
                    break;
        case 'P':  cin >> polynum;
              cout << Polynomials[polynum];
               break;
        default: cout << "I missed something" << endl;
  }
  cin >> command;
}
```

Output: After the operations A or M print the resultant polynomial in the sorted order of the exponent. After the operations I and D print the success or failure of the operation. After the E operation print the value of the polynomial. The P command prints the following format.
For example, $2x^6 + 10x^3 – 2x + 4$ will be printed as the following when it hits the command "**`P 1`**":

```
Polynomial 1: (2,6) + (10,3) + (-2,1) + (4,0)
```

Constraints:

- In this project, the only header you will use is #include <iostream>.
- None of the projects is a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.