National University of Computer and Emerging Sciences, Islamabad Campus

Course:	**Data Structures**	**Course Code:**	**CS 2001**
Program:	**BS(Computer Science)**	**Semester:**	**Spring 2022**
Due Date	**27-Mar-2022 at 11:59 pm**	**Total Marks:**	**20**
Type:	**Assignment 1**	**Page(s):**	**3**

Important Instructions:

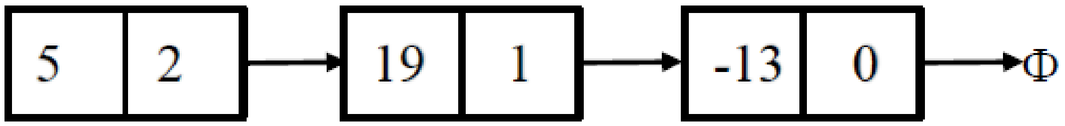
1. Submit your source files in a zipped file named as your roll number, i.e., 20I-1111.zip.
2. You are not allowed to copy solutions from other students. We will check your code for plagiarism using plagiarism checkers. If any sort of cheating is found, negative marks will be given to all students involved.
3. Late submission of your solution is not allowed.

Introduction:

We want to design a math solver that can solve large polynomials. Mathematically a polynomial is a collection of terms where each term has a variable exponent and a coefficient. For simplicity, we can assume that exponent is a non-negative integer and coefficient can be any real number. For example P(n) = $5n^2 + 19n - 13$ is a polynomial with three terms: $5n^2$, 19n and -13. First term has coefficient 5 and exponent 2, second term has coefficient 19 and exponent 1. The last term has coefficient -13 and exponent 0. Various operations can be defined on polynomials that includes addition, multiplication and evaluation. Addition and multiplication are binary operations where both operands are polynomials that add or multiply two polynomials and a resultant polynomial is generated. However, evaluate is a binary operator that takes a real value for n and evaluates P(n). For example P(2) = 5(4) + 19(2) -13 = 45.

Implementation

Your task is to design a polynomial calculator that can perform three basic operations on large polynomials: add, multiply, evaluate. Since number of terms in a polynomial can vary, we can borrow the idea of linked list to efficiently implement polynomials. Each polynomial must have the address of the first term and each term except the last term has the address of the next term. You can also use a variable size to store the total number of terms in a polynomial. Below in an example of this structure.



IMPORTANT CLASSES
You have to implement the following classes

Class Term
This class must have following members:

- Exponent
- Coefficient
- Pointer to term

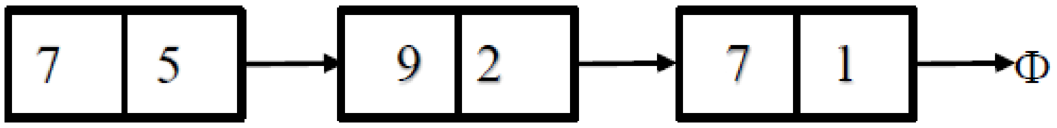
Implement this class as nested class of polynomial

Class Polynomial
This class must implement the following data members and member functions:
Data Members:

- Pointer to term
- Size

Member function:

Input(): This function must input the terms of polynomial and place these terms in the sorted order of the exponent (decreasing order). First take the number of terms as input and then take the coefficient and exponent of each term as input. Implement all the appropriate checks and make sure that polynomial is in its simplest form is stored in memory (there can be at most one term for a unique exponent). For example, if the user wants to input the polynomial $5n + 9n^2 - 12n + 7n^5$ having four terms then it must be stored as $7n^5+9n^2-7n$. Below is the linked form of the stored polynomial.



Output(): This function must output the polynomial in the desired format as mentioned below:
Output Format of a polynomial:

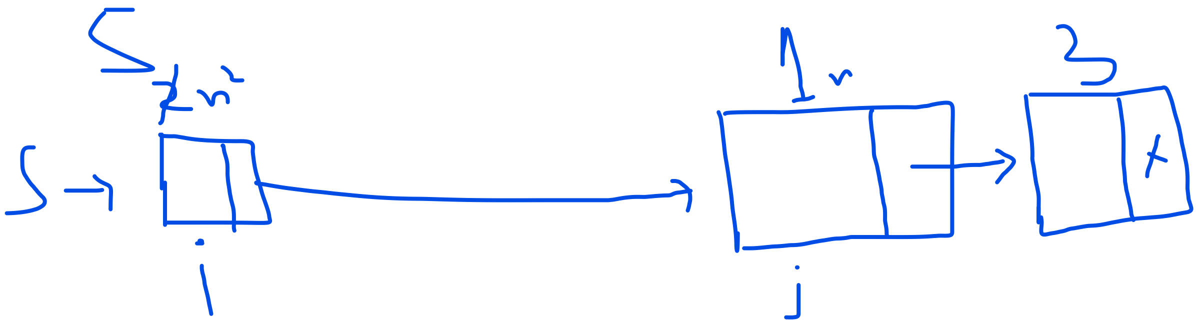
- A term with coefficient c and exponent x must be displayed as cn^x
- If a term is not the first term, then the sign of the coefficient (+/-) must be displayed.
- If the exponent of a term is 0 then do not output the variable 'n'

Add(): This function must take a polynomial as parameter, add it with the caller object and return the resultant polynomial. Use the standard addition procedure for polynomials. Also overload the '+' operator for this operation. This operation must be done in O(n+m) time where n and m are the number of terms in polynomials to be added

Multiply(): This function must take a polynomial as parameter, multiply it with the caller object and return the resultant polynomial. Use the standard multiplication procedure for polynomials. Also overload the '*' operator for this operation. This operation must be done in O(n*m) time where n and m are the number of terms in polynomials.

Evaluate(): This function takes a real number as input, evaluate the value of the polynomial and return it. Overload the '.' Operator for this operation.

Also implement default constructor, copy constructor, destructor, overloaded assignment operator (implements deep copy) or any other function that you think is required.


5
2
n^2
s →
i
1
n
j
3
x