OOAIA Lab 10

Ramanujan and Polynomials

Problem statement:

In a small town nestled between rolling hills and lush forests, there lived a brilliant mathematician named Ramanujan. He had recently been approached by the town's mayor to help with various projects that required mathematical modeling and analysis. The town was planning to expand its infrastructure, which included building new roads, parks, and community centers. To ensure efficient resource allocation and budgeting, Ramanujan needed to develop mathematical models using polynomials. These models would help predict growth, optimize resource usage, and analyze financial trends.

Tasks:

i. **Polynomial Multiplication**:

a. Develop an algorithm that can multiply two polynomials efficiently (you are expected to design a divide and conquer algorithm for polynomial multiplication). You can refer to this and this for algorithms related to the divide and conquer approach of polynomial multiplication.
b. The algorithm should support different data types for polynomials:
   - Integers (Use long long int for this)
   - Floats (use long double for this)
   - Complex numbers (use long long int for the real and imaginary values)
c. Overload the `*` operator for polynomial objects.

ii. **Polynomial evaluation**

a. Create a algorithm to evaluate polynomials at a given value of x
b. The algorithm should support different data types for polynomials:
   - Integers (Use long long int for this)

- Floats (use long double for this)
- Strings (operations on strings mean string concatenation)

iii. **Polynomial differentiation**

a. Develop an algorithm to differentiate polynomials. The differentiation should be restricted to one degree.
b. The algorithm should support different data types for polynomials:
   - Integers (Use long long int for this)
   - Floats (use long double for this)

Representation of the polynomials in input and output:

Each polynomial is represented by the coefficients of increasing degree in the input and output.
For example,

the polynomial $ax^3 + bx^2 + cx + d$ will be represented in the input as (d, c, b, a)
The polynomial $px^4 + qx$ will be represented in the output as (0, q, 0, 0, p)

Examples of each operation:

i. **Polynomial Multiplication**:

Data Type	Polynomial 1	Polynomial 2	Output	Explanation
Integer (with only positive coefficients)	{3, 2, 5} this is $3 + 2x + 5x^2$	{5, 1} this is $5 + x$	{15, 13, 27, 5} this is $15 + 13x + 27x^2 + 5x^3$	When we multiply ($3 + 2x + 5x^2$) and (5 + x), we get ($5x^3 + 25x^2 + 2x^2 + 10x + 3x + 15$) which simplifies to ($15 + 13x + 27x^2 + 5x^3$)
Integer (with positive and negative coefficients)	{3, -2, 5} this is $3 - 2x + 5x^2$	{-5, 1} this is $-5 + x$	{-15, 13, -27, 5} this is $-15 + 13x - 27x^2 + 5x^3$	Same
Floats (with only	{3.5, 2.1, 5.7}	{1.2, 3.8} this is 1.2 +	{4.2, 15.82, 14.82, 21.66}	Same

positive coefficients)	this is $3.5 + 2.1x + 5.7x^2$	$3.8x$	this is $4.2 + 15.82x + 14.82x^2 + 21.66x^3$	
Floats (with positive and negative coefficients)	{3.5, -2.1, 5.7} this is $3.5 - 2.1x + 5.7x^2$	{-1.2, 3.8} this is $-1.2 + 3.8x$	{-4.2, 15.82, -14.82, 21.66} this is $-4.2 + 15.82x - 14.82x^2 + 21.66x^3$	Same
Complex number (with positive and negative coefficients)	{(1, -2), (3, 4), (-5, 6)} this is $(1 - 2i) + (3 + 4i)x + (-5 + 6)x^2$	{(-7, 8), (9, -10)} this is $(-7 + 8i) + (9 - 10i)x$	{(9, 22), (-64, -32), (54, -76), (15, 104)} this is $(9 + 22i)+ (-64 - 32i)x + (54 - 76i)x^2 + (15 + 104i)x^3$	When we multiply $\{(1 - 2i) + (3 + 4i)x + (-5 + 6i)x^2\}$ and $\{(-7 + 8i) + (9 - 10i)x\}$ we get $(-7 + 8i + 14i - 16i^2) + (9 -10i -18i + 20i^2)x + (-21 + 24i - 28i + 32i^2)x + (27 - 30i + 36i - 40i^2)x^2 + (35 -40i - 42i + 48i^2)x^2 + (-45 + 50i + 54i - 60i^2)x^3$ which simplifies as $(9 + 22i) + (-11 -28i)x + (-53 - 4i)x + (67 + 6i)x^2 + (-13 -82i)x^2 + (15 + 104i)x^3$ which ultimately simplifies to $(9 + 22i) + (-64 -32i)x + (54 - 76i)x^2 + (15 + 104i)x^3$

ii. **Polynomial Evaluation**:

Data Type	Polynomial 1	Value of x	Output	Explanation
Integer	{7, -5, 0, 3} this is $7 - 5x + 3x^3$	2	21	The polynomial $7 - 5x + 3x^3$ evaluates to $7 - 5 * 2 + 3 * 2^3 = 21$
Float	{7, -5.2, 0, 3} this is $7 - 5.2x + 3x^3$	2	20.6	The polynomial $7 - 5x + 3x^3$ evaluates to $7 - 5.2 * 2 + 3 * 2^3 = 20.6$

String	{“7”, “-5”, “6”, “abc”} this is “7” + “- 5”x + “6”x^2 + “abc”x^3	2	abcabcabcabc abcabcabcabc 6666-5-57 The output string is concatenated in the increasing order of the degree (i.e. the string with x^3 should be present before the string with x^2)	When we evaluate the given polynomial at x = 2, we get “7” + “-5” * 2 + “6” * 4 + “abc” * 8 Now in this expression, the integer means the #times the preceding string should be concatenated. Hence we get the output abcabcabcabcabcabcabca bc6666-5-57

iii. **Polynomial Differentiation**:

Data Type	Polynomial	Output	Explanation
Integer	{7, -5, 0, 3} this is 7 - 5x + $3x^3$	{-5, 0, 9} this is -5 + $9x^2$	When we differentiate the polynomial 7 - 5x + $3x^3$ by one degree then we get the resultant polynomial as -5 + $9x^2$
Float	{7, -5.2, 0, 3.5} this is 7 - 5.2x + $3.5x^3$	{-5.2, 0, 10.5} this is -5.2 + $10.5x^2$	Same

Design Requirements:

Each of the following carries equal weightage of 0.3 marks for the total 1.5 marks in design

1. Design classes representing each of the three operations separately (multiplication, differentiation, evaluation).
2. Create templates for each of the class to support the different data types for each operation.
3. Use a divide and conquer approach for the multiplication operation.
4. Overload the operator * for the multiplication operation.
5. Suggestive comments or self explanatory variable and function names.

Test cases format:

Each test case may contain as many queries for the operations.

Input format:

- The first line contains an integer q denoting the number of queries
- Then follows q queries with the following format:
    - Each query starts with an integer op denoting the type of operation (1 denotes multiplication, 2 denotes evaluation and 3 denotes differentiation)
    - For multiplication operation:
        - The first line contains one string from ["integer", "float", "complex"] denoting the data type of the coefficients of the polynomials (both the polynomials will have the same data type for coefficients)
        - The next line contains one integer deg1 denoting the number of coefficients for the first polynomial
        - If the data type is integer or float:
            - The next line contains deg1 space separated values (of the data type integer of float accordingly) denoting the coefficients of the first polynomial (in increasing degree, i.e. starting from the coefficient of x^0 to the coefficient of $x^{deg1 - 1}$)
        - If the data type is complex:
            - The next line contains 2*deg1 space separated integers. In each pair of the 2*deg1 values, the first value denotes the real part and the second value denotes the imaginary part. This together form the coefficient of one degree and similarly, the pairs of coefficients are given for the deg1 degrees of the polynomial (in increasing degree, i.e. starting from the coefficient of x^0 to the coefficient of $x^{deg1 - 1}$).
        - The next line contains one integer deg2 denoting the number of coefficients for the second polynomial

        - If the data type is integer or float:
            - The next line contains deg2 space separated values (of the data type integer of float accordingly) denoting the coefficients of the first polynomial (in increasing degree, i.e. starting from the coefficient of x^0 to the coefficient of $x^{deg2 - 1}$)
        - If the data type is complex:
            - The next line contains 2*deg2 space separated integers. In each pair of the 2*deg2 values, the first value denotes the real part and the second value denotes the imaginary part. This together form the coefficient of one degree and similarly, the pairs of coefficients are given for the deg2 degrees of the polynomial (in increasing degree, i.e. starting from the coefficient of x^0 to the coefficient of $x^{deg2 - 1}$).
    - For evaluation operation:
        - The first line contains one string from ["integer", "float", "string"] denoting the data type of the coefficients of the polynomial
        - The next line contains one integer deg denoting the number of coefficients for the polynomial
        - The next line contains deg space separated values (of the data type mentioned above) denoting the coefficients of the polynomial (in increasing degree, i.e. starting from the coefficient of x^0 to the coefficient of $x^{deg - 1}$)
        - The next line contains one integer denoting the value of x at which the polynomial should be evaluated
    - For differentiation operation:
        - The first line contains one string from ["integer", "float"] denoting the data type of the coefficients of the polynomial
        - The next line contains one integer deg denoting the number of coefficients for the polynomial
        - The next line contains deg space separated values (of the data type mentioned above) denoting the coefficients of the polynomial (in increasing degree, i.e. starting from the coefficient of x^0 to the coefficient of $x^{deg - 1}$)

Output format:
For the outputs of floats, print the values up to 6 decimal places.

- For each query there will be one line of output.
- For multiplication operation:

    - One line of space separated values denoting the coefficient of the resulting polynomial. If the resulting polynomial is of degree deg:
        - For integer or float data type:
            - There should be deg space separated values. For example, if the resultant polynomial is $bx + cx^3$ then the output should be

0 b 0 c

        - For complex data type:
            - There should be 2*deg separated integers (one pair denoting the coefficient of each degree, where the real part comes first and the imaginary part comes next). For example, if the resulting polynomial is $(a + ib) + (c + id)x^2$ then the output should be

a b 0 0 c d

- For evaluation operation:
    - One single value representing the evaluated value of the polynomial.
- For differentiation operation:
    - One line of space separated values denoting the coefficient of the resulting polynomial.

Sample test case:

Sample input:

3
1
complex
3
1 -2 3 4 -5 6
2
-7 8 9 -10
2
string
4
7 -5 6 abc
2
3
integer

4
7 -5 0 3

Sample output:
9 22 -64 -32 54 -76 15 104
abcabcabcabcabcabcabcabc6666-5-57
-5 0 9

Explanation:
There are 3 queries in the test case.

The first query is of type multiplication (because query type number is 1).
The data type is complex
The first polynomial has 3 pairs of coefficients.
The pairs are (1, -2), (3, 4) and (-5, 6). So the polynomial is $(1 - 2i) + (3 + 4i)x + (-5 + 6i)x^2$
The second polynomial has 2 pairs of coefficients.
The pairs are (-7, 8) and (9, -10). So the polynomial is $(-7 + 8i) + (9 - 10i)x$.
The result of the multiplication is $(9 + 22i)+ (-64 - 32i)x + (54 - 76i)x^2 + (15 + 104i)x^3$ (see the table of examples for the explanation).
So the output is

9 22 -64 -32 54 -76 15 104

The second query is of type evaluation (because query type number is 2).
The data type is string
The polynomials have 4 coefficients.
These are “7”, “-5”, “6” and “abc”. So the polynomial is “7” + “-5”x + “6”x^2 + “abc”x^3
The value of x which the polynomial should be evaluated is 3.
The result of the evaluation is abcabcabcabcabcabcabcabc6666-5-57 (see the table of examples for the explanation).
So the output is

abcabcabcabcabcabcabcabc6666-5-57

The third query is of type differentiation (because query type number is 3).
The data type is integer
The polynomials have 4 coefficients.
These are 7, -5, 0 and 3. So the polynomial is $7 - 5x + 3x^3$
The resultant polynomial after differentiation is $-5 + 9x^2$ (see the table of examples for the explanation).
So the output is

-5 0 9

Constraints:

Number of queries in each test case <= 1500

-10^4 <= Range of values for integer and floats <= 10^4 - 1

Length of string for each coefficient <=50

1 <= Value of x (value at which evaluation has to be done) <= 10

1 <= Maximum degree of the polynomial <= 10

Notes:

- **It is advised to use long long int for storing the integer values and long double for storing the float values. This is to maintain the precision, prevent any overflow and avoid any incorrect output due to smaller data types.**
- The output of float values should be up to 6 decimal places.
  You can use the following statement to print up to 6 decimal places.
  cout << fixed << setprecision(6) << var;
- For the evaluation operation, if the polynomial is of type string then all the coefficients should be treated as a string (even if the coefficient is a digit).
- For evaluation operation, the value of x at which the polynomial has to be evaluated will always be an integer.
- For evaluation operation of strings, the resultant string is formed from higher degree to the lower degree (i.e. first the string with x^{n-1} is present followed by the lower degrees up to x^0).
- You are advised to use divide and conquer algorithms for the multiplication operation.
- **You are expected to design our own complex class for the support of complex numbers. Do not do a #include<complex>**