Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Programming and Data Structures, 2023-24, Semester-II
Assignment 05

Maximum Marks: **150** Submission Deadline: **2023-Oct-....**
Topic: Polynomial Operations

Assignment problem # AP0501

Polynomial operations: Given two polynomials $f(x) = \sum_{i=0}^{n} a_i.x^i$, $g(x) = \sum_{i=0}^{m} b_i.x^i$, denoted by A and B, of degree n and m respectively, find addition/ subtraction/division/multiplication of them. Assume that a polynomial is represented by the structure `typedef struct poly { float * Coef; int deg;} Poly;` that stores degree of a polynomial and coefficient array. You should have at least the following operations.

- `Poly` ← `allocate_n__init`(n): Given a non-negative integer n, it initializes a polynomial structure A. Here it allocates memory for the coefficients for a polynomial of degree n and finally returns A. Consider the coefficients as `float` variables.
- b ←`poly_display`(`Poly` A): Given a polynomial A, it should display the polynomial. Output should be in such a way that all of your friends can understand. Finally it returns a status bit b (b = degree of the polynomial if success, else return -1, in case of failure). The coefficients must be displayed up to 2 decimal places.
- b ←`poly_free`(`Poly` A): Given a polynomial A, it makes the memory allocated for the coefficients free. Finally, it returns a status bit b (b = degree of the polynomial if success, else return -1, in case of failure).
- `Poly` ← `poly_add`(`Poly` A, `Poly` B): Given two polynomials `Poly` A and `Poly` B, it outputs a polynomial $C = A + B$ and displays C in the terminal.
- `Poly` ← `poly_sub`(`Poly` A, `Poly` B): Given two polynomials A and B, it outputs a polynomial $C = A - B$ and displays C in the terminal
- `Poly` ← `poly_mult`(`Poly` A, `Poly` B): Given two polynomials A and B, it outputs a polynomial $C = A * B$ and displays C in the terminal.
- `PoliDivRes` ← `poly_div`(`Poly` A, `Poly` B): Given two polynomials A and B, it outputs `PoliDivRes` which stores a polynomial R (remainder) and a polynomial Q(quotient) such that $A = B * Q + R$ and displays R and Q in the terminal
- `Poly` ← `poly_mult_dnc`(`Poly` A, `Poly` B): Given two polynomials A and B, it computes a polynomial $C = A * B$ using divide and conquer method and displays C in the terminal.

Input format: A file containing $(3k + 1)$ lines.

- Line 1 contains the number of test cases, i.e., k.
- Each test case has three lines:
  1. line 1 contains n m op, separated by space, where $op \in \{+, -, *, /\}$, and n and m are degrees of the input polynomials.
  2. line 2 contains space separated coefficients of the 1st polynomial with degree n as $a_n a_{n-1} \ldots a_0$.
  3. line 3 contains space separated coefficients of the 2nd polynomial with degree m similar to the above.

Output format: Any Readable format.

Notes: Free the memories occupied by the polynomials, if any, before terminating the program at any stage.

Marks Distribution: Main code: 90+30; Good programming practices: 30