Introduction to Computer Programming and Data Structures
Assignment 05

Maximum Marks: **150** Submission Deadline: **2023-May-06**
Topic: Polynomial Operations

Assignment problem # AP0501

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

- $poly\_A \leftarrow$ `poly_init`(n): Given a non-negative integer n, it initializes a polynomial structure $poly\_A$. Here it allocates memory for the coefficients and the degree in $poly\_A$. Consider the coefficients as `float` variables.
- $b \leftarrow$`poly_display`$(poly\_A)$: Given a polynomial $poly\_A$, 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.
- $b \leftarrow$`poly_free`$(poly\_A)$: Given a polynomial $poly\_A$, 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).
- $poly\_C \leftarrow$ `poly_add`$(poly\_A, poly\_B)$: Given two polynomials $poly\_A$ and $poly\_B$, it outputs $poly\_C = poly\_A + poly\_B$ and displays $poly\_C$ in the terminal.
- $poly\_C \leftarrow$ `poly_sub`$(poly\_A, poly\_B)$: Given two polynomials $poly\_A$ and $poly\_B$, it outputs $poly\_C = poly\_A - poly\_B$ and displays $poly\_C$ in the terminal
- $poly\_C \leftarrow$ `poly_mult`$(poly\_A, poly\_B)$: Given two polynomials $poly\_A$ and $poly\_B$, it outputs $poly\_C = poly\_A * poly\_B$ and displays $poly\_C$ in the terminal.
- $PoliDivRes \leftarrow$ `poly_div`$(poly\_A, poly\_B)$: Given two polynomials $poly\_A$ and $poly\_B$, it outputs $PoliDivRes$ which stores $poly\_R$ (remainder) and $poly\_Q$(quotient) such that $poly\_A = poly\_B * poly\_Q + poly\_R$ and displays $poly\_R$ and $poly\_Q$ 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. For multiplication, output results from both `poly_mult` and `poly_mult_dnc`.

Notes:

- $poly\_A, poly\_B, poly\_C$, etc., are structures that store polynomials (i.e., store the coefficient and degree).
- Free the memories occupied by the polynomials, if any, before terminating the program at any stage.

[15+15+15+15+15+25+30+20]