MATLAB Programming NYCU Fall 2022 Assignment#2 due 2022/11/5

In this assignment, you will write a function file with several local functions. The function file is used to do basic operations on polynomials. Here we use a **Nx2** array to represent a polynomial of **N** terms. For example:

Array: `[3 1; 2 5; 0 -2]` ➔ Polynomial: $x^3 + 5x^2 - 2$
Array: `[100 2; 10 3]` ➔ Polynomial: $2x^{100} + 3x^{10}$

Rules of the representation are:

- Each row represents one term of the polynomial in the form of `[degree coefficient]`.
- The terms are in descending order of degrees.
- No two terms can have the same degree.
- Only terms with nonzero coefficients are included. The only exception is when the polynomial is zero, in which case it contains only one row `[0 0]`.

The main function of your function file (let's call it `func` for now) should handle the following forms of inputs:

- **`poly_out = func(op, poly1, poly2)`**

  Here `poly1` and `poly2` are input polynomials, and `poly_out` is the output polynomial given by the operation specified by `op`. Here `op` should be a character vector being one of the following: `'add'`, `'subtract'`, and `'multiply'`.

- **`Y = func(op, poly1, X)`**

  In this form, only one input polynomial is supplied. Here `op` should be `'plot'` or `'eval'`. For both `op`, `X` is a numeric array of x values used to evaluate the polynomial, and output `Y`, containing the evaluation results. is a numeric array of the same size of `X`. If `op` is `'plot'`, `X` needs to be a sorted vector, and a plot of (`X`,`Y`) should be generated.

In your function file, you need to implement the five local functions (`poly_add`, `poly_subtract`, `poly_multiply`, `poly_eval`, `poly_plot`) for the five given operations.

Some notes and hints:

- If `poly_out` is returned, it is important that it satisfies the rules of the representation stated above. For example, zero-coefficient terms should be removed, and terms of the same degree should be merged. (Although not required, you can add an extra local function just for making a polynomial consistent with the rules. This extra function can be called by `poly_add`, `poly_subtract`, and `poly_multiply` to process their direct results.)
- Inside `poly_plot`, you can just call `poly_eval` to compute the values for plotting.
- In the main function, check `op` first before checking other inputs. Apply `validatestring` to `op` first. Then use the `switch-case` statement to handle the processing of different `op`.
- There are a lot of input checking that you should do. For example, the input polynomials should have shape of **Nx2** and the first column should contain only nonnegative integers. I will not list all the input checking that are applicable here. Try to think carefully and be as complete as possible.

Submission: Submit your code (m file) through E3. Name your file `P2_#######.m`, where the `#######` represents your student ID. (`P2_#######` should be the name of your main function.) There will be a three-day grace period after the due date, during which there will be a 10%/day deduction for your grade.

A "copy detection" will be applied to your submissions, and those found to have copied assignments will receive zero points for the assignment.

Your code should include sufficient comments. This will be part of the grade. Include your name and ID at the top of your code.

There will be demo session with the TAs (date/time to be announced later).