COSE213-01 hw1

2018320192 손재민

1. Overview

- This program gets two polynomials and calculate the sum of two polynomials. Program gets polynomials by two strings of coefficients and exponents separated by spaces. It is designed to work for the descending sorted input. Improper input is rejected.
- At the start of the program, it gets two polynomials by input. Then add these polynomials and print the result. Input and result polynomials would be stored on the program.

2. Variables

- MAX_TERMS : Max number of terms.
- terms[MAX_TERMS] : An array used to store terms. Max size of terms is under MAX_TERMS.
- avail : Indicate the end of the terms array. New polynomials would be start at avail position. At the start of the program it is initialized as 0.
- startA, endA, startB, endB, startR, endR: Start and end positions of polynomial A, B, and R.

3. Functions

- void printPoly(int startPoly, int endPoly) : Get the start position of polynomial and end position of polynomial in terms array and print all polynomials by the form of (coefficient) x ^ (exponent).
- void attachPoly(float coef, int exp) : Get coefficient and exponent as inputs and add them in terms array, increasing avail. If avail exceeds MAX_TERMS(if the number of terms exceeds max size of terms array), then return error.
- void newPoly(int* startPoly, int* endpoly) : Get two pointer as a start position of a polynomial and end position of a polynomial. Start position is set as avail at the start of the function. Then get coefficients and exponents until input is an enter('₩n'). Function returns error if coefficient is not real number, exponent is not integer over 0, or coefficient and exponent do not paired(if amount of input number is odd). End position is set as avail – 1 at the end of the function.
- void addPoly(int startA, int endA, int startB, int endB, int* startR, int* endR) : Get two polynomials' start and end positions and pointer of result polynomial's start and end

position. Start position of result polynomial is set as avail at the start of the function. Then check polynomial A and B and add each terms from start position until any one of polynomial reaches end. If one's exponent is bigger than the other's, then add bigger one to the terms array and move bigger one to next position. If both exponents are same, then add new term by their exponents for exponent and sum of two coefficients for coefficient, and move A and B to the next position. If any polynomial reaches to the end, get out from the loop and add remaining terms to the terms array. At the end of the loop, there would remain only one polynomial or no polynomial, so it is okay to add terms in the original order. End position of result polynomial is set as avail – 1 at the end of function.