# 실습 3 – 다항식의 덧셈과 곱셈(배열)

- 실습 목표
  - 구조체 배열의 개념과 동적 메모리 할당을 이해하고 구현할 수 있다.
  - 다항식을 구조체로 표현하고, 이들의 합과 곱을 구할 수 있다.

# 다항식 생성

- **(3.1.)** 다항식의 구조 (main 함수)
  - 항을 (계수-실수, 지수-정수) 쌍의 구조체(polynomial)로 표현
  - polynomial 구조체 배열을 동적으로 생성(처음에는 10개의 항을 갖도록)
    - struct polynomial *A, *B, *D;
  - 배열에는 항들이 지수의 내림차순으로 저장
    - 제일 마지막 항은 다항식의 끝을 표시하기 위하여 지수를 -1로 지정
  - 다항식 생성 방법
    - 방법 1: 각 항의 계수와 지수를 입력받아 저장
    - 방법 2: 무작위로 계수를 생성하여 저장
    - 각 항은 지수의 내림차순으로 정렬되어야 함
  - 만약 항의 수가 10보다 커지면, realloc을 이용하여 배열 크기를 증가

# 다항식의 합

- **(3.2.)** 프로그램 2.6과 2.7 구현
  - 함수원형 :
    polynomial* padd(polynomial* A, polynomial* B)
    void attach(float cofficient, int exponent)
  - 배열의 구조가 바뀌었으므로, 일부 수정 필요
  - 다항식의 합을 polynomial *D에 저장한 후, D를 반환

- main 함수에서 다항식 A와 B를 생성한 후, D = padd(A,B)를 호출하고, 그 결과를 확인
  - 결과 확인을 위하여 다항식을 출력하는 함수를 작성

# 다항식의 곱

- **(3.3.)** 앞에서 구현한 padd를 이용하여, 두 다항식을 곱하는 함수 pmul을 작성

- D = pmul(A, B)의 경우
  - 함수원형 :
    polynomial* pmul(polynomial* A, polynomial* B)
  - A의 항 하나와 B의 다항식을 곱하는 Ci = single_mul(A[i], B)를 구현
    - 함수원형 :
      polynomial* single_mul(polynomial Ai, polynomial* B)
  - D = padd(D, Ci)를 이용하여 Ci를 D에 추가
  - A의 모든 항에 대해 이 과정을 반복

- 실습 시험 예상 문제 중 하나임.