[addPoly.h]

```
#pragma once  // 컴파일러에게 해당 헤더 파일이 한 번만 빌드되도록 하여 여러 번 include 되는 것을 막아준다.
#define MAX(a, b) ((a > b) ? a : b)  // a와 b를 비교하여 더 큰 값을 반환
#define MAX_DEGREE 50  // MAX_DEGREE의 값을 기호상수를 이용해 50으로 정의

typedef struct  // 구조체 polynomial 정의
{
    int degree;  // 다항식의 차수를 저장할 정수형 변수 degree 선언
    float coef[MAX_DEGREE];  // 다항식의 각 항의 계수를 저장할 1차원 실수형 배열 coef 선언 (크기: MAX_DEGREE)
} polynomial;  // 구조체 별칭

polynomial addPoly(polynomial A, polynomial B);  // polynomial A와 polynomial B를 더한값을 리턴해주는 함수 호출
void printPoly(polynomial P);  // polynomial P를 출력하는 함수 호출.
```

[addPoly.c]

```
#include "addPoly.h"

polynomial addPoly(polynomial A, polynomial B)  // 다항식 A와 B를 매개변수로 받아 덧셈을 수행하는 addPoly 함수
{
    polynomial C;  // 다항식 덧셈의 결과를 저장할 polynomial 구조체 변수 C 선언

    int A_index = 0, B_index = 0, C_index = 0;  // 배열 coef를 인덱스별로 처리하기 위한 정수형 변수 선언 및 0으로 초기화
    int A_degree = A.degree, B_degree = B.degree;  // 다항식 A, B의 차수를 저장할 정수형 변수
    C.degree = MAX(A.degree, B.degree);
      // A의 차수와 B의 차수를 비교하여 더 큰 값을 C의 차수로 저장 (선언 및 초기화)
```

```
while (A_Index <= A.degree && B_Index <= B.degree) // 두 다항식에서 처리할 항이 없을 때까지 반복
{
    if (A_degree > B_degree) // 다항식 A의 항의 차수가 다항식 B의 항의 차수보다 크면 실행
    {
        C.coef[C_Index++] = A.coef[A_Index++]; // 현재 다항식 A의 항의 계수를 다항식 C의
        A_degree--;  // 다음에 처리할 항을 위해 감소                현재 항의 계수로 저장
    }
    else if (A_degree == B_degree) // 다항식 A와 B의 항의 차수가 같은 경우 실행
    {
        C.coef[C_Index++] = A.coef[A_Index++] + B.coef[B_Index++]; // 다항식 A와 B의
        A_degree--; // 다음에 처리할 항을 위해 감소                  계수를 더하여 다항식 C의
        B_degree--; // 다음에 처리할 항을 위해 감소                  현재 항의 계수로 저장
    }
    else. // 다항식 B의 항의 차수가 다항식 A의 항의 차수보다 크면 실행
    {
        C.coef[C_Index++] = B.coef[B_Index++]; // 현재 다항식 B의 항의 계수를 다항식 C의
        B.degree--; // 다음에 처리할 항을 위해 감소                  현재 항의 계수로 저장
    }
}
return C;  // 다항식 덧셈의 결과 C를 반환
}
void printPoly(polynomial P) // polynomial P를 출력하는 함수 선언
{
    int degree;  // 차수를 저장할 정수형 변수 선언
    degree = P.degree;  // P를 매개변수로 받아 차수를 degree에 저장
```

NO.
year month day ( )

```
    for (int i = 0; i <= P.degree; i++) // 다항식의 최고차항까지 반복
    {
        printf("%3.0fx^%d", P.coef[i], degree--); // 다항식 출력
        if(i < P.degree)  // 차수가 남았다면 실행
            printf(" +");  // 차수가 남아있다면 두 차수를 더해야 다항식이 되므로 '+' 출력
    }
    printf("\n");  // 하나의 다항식이 끝났을 경우 줄바꿈
}
```

[Ex3_6.c]

```
#include <stdio.h> // 표준입출력에 관한 함수들이 정의되어 있는 헤더파일 포함
#include "addPoly.h" // addPoly.h 헤더 파일 포함

int main()
{
    polynomial A = {3, {4, 3, 5, 0}};  // 다항식 A의 초기화
    polynomial B = {4, {3, 1, 0, 2, 1}};  // 다항식 B의 초기화
    polynomial C;  // 다항식 C에는 A와 B를 더한 값이 들어가야 하므로 선언.
    C = addPoly(A, B);  // addPoly 함수를 호출하여 다항식 A, B에 대한 덧셈을 수행

    printf("\n A(x) = "); printPoly(A);  // printPoly 함수를 호출하여 다항식 A 출력
    printf("\n B(x) = "); printPoly(B);  // printPoly 함수를 호출하여 다항식 B 출력.
    printf("\n C(x) = "); printPoly(C);  // printPoly 함수를 호출하여 다항식 C 출력
    getch();  // 버퍼를 사용하지 않고 한 글자 입력받기.
    return 0;  // main 함수를 성공적으로 끝낸 후 종료
}
```