실습과제 2.3

[과제 설명]

다항식(한 개 이상의 항의 합으로 이루어진 식) 두 개 A, B가 주어졌을 때, 두 개의 다항식의 합, 다항식 C를 출력한다.
이 때 구조체형 배열을 사용하여 자료구조를, 배열 기반의 다항식의 덧셈 함수를 이용한다. 구조체에는 (계수 coef, 차수 expon)의 쌍으로 표현한다.

다항식 덧셈 함수에서는 두 다항식의 차수를 비교하고, 두 다항식의 차수가 동일한 경우에는 (더한 값이 0이 아닌 경우에 한해서)더해서 저장하고, 두 다항식의 차수가 다르면 차수가 큰 쪽부터 순차적으로 저장하고 각 경우에 저장하는 현재 위치를 표시하는 avail을 1씩 증가시켜준다.

마지막의 잔여항은 A와 B가 각각 몇 개가 남아있는지 모르기 때문에 옮겨진 시작지점(As, Bs)부터 끝지점(Ae, Be)까지 반복문을 실행해서 끝까지 더해준다.

[코드]

```
#include <stdio.h>

#define NO_OF_TERMS 20
//A, B, C의 항 개수의 합 이상으로(여유분 포함)

typedef struct {
  int coef; //계수
  int expon; //차수
}Term;

int Cs;
int Ce;

void insert(Term poly[], int avail, int coef, int expon) {
  poly[avail].coef = coef;
  poly[avail].expon = expon;
  //avail의 위치에 계수와 차수를 대입
}

void Poly_Add(Term poly[], int As, int Ae, int Bs, int Be, int* avail) {
  Cs = *avail; //초기 avail의 값이 C의 시작 위치가 된다.
```

```
  while (As <= Ae && Bs <= Be) { //A와 B 항의 차수 비교
    if (poly[As].expon > poly[Bs].expon) {//A의 차수가 큰 경우
      insert(poly, *avail, poly[As].coef, poly[As].expon);
      As++; //비교할 위치를 뒤로 한 칸 옮겨준다.
    }
    else if (poly[As].expon == poly[Bs].expon) {//차수가 같은 경우
      int sumCoef = poly[As].coef + poly[Bs].coef;

      if (sumCoef) { //차수가 같은 항의 계수합이 0이 아닌 경우
        insert(poly, *avail, sumCoef, poly[As].expon);
        As++;
        Bs++;
      }
      else (*avail)--;
      //0인 경우에 avail의 위치를 1 감소시킨다(마지막에 항상 1 증가하기 때문에).
    }
    else { //B의 차수가 큰 경우
      insert(poly, *avail, poly[Bs].coef, poly[Bs].expon);
      Bs++;
    }

    (*avail)++;//avail의 위치를 1 증가시킨다.
    //포인터기 때문에 단순히 avail++은 불가능
  }

  for (int i = As; i <= Ae; i++) { //A의 잔여항 대입
    insert(poly, *avail, poly[As].coef, poly[As].expon);
    (*avail)++;
  }
  for (int i = Bs; i <= Be; i++) { //B의 잔여항 대입
    insert(poly, *avail, poly[Bs].coef, poly[Bs].expon);
    (*avail)++;
  }

  Ce = *avail - 1;
  //avail은 마지막 부분에서 1 증가하고 끝났기 때문에
  //C의 마지막 위치는 avail에서 1을 빼주어야 한다.
}

void Print_Poly (Term poly[], int As, int Ae, int Bs, int Be, int avail) {
//다항식 출력함수
  for (int i = 0; i < avail; i++) {
    if (i == As) printf("Polynomial A :  ");
    if (i == Bs) printf("Polynomial B :  ");
    if (i == Cs) printf("Polynomial C :  ");

    if(poly[i].expon != 0) //차수가 0이 아닐 때 차수와 함께 출력
      printf("%dx ^ %d", poly[i].coef, poly[i].expon);
    else //차수가 0이면 계수만 출력
      printf("%d ", poly[i].coef);

    if (i == Ae || i == Be || i == Ce) puts("");
    else printf(" + "); //마지막이 아니면 + 같이 출력
  }
}

int main(void) {
```

```
  Term poly[NO_OF_TERMS] = {{3,5}, {2,3}, {2,0}, {4,4}, {5,3}, {2,1}};
  int As = 0, Ae = 2, Bs = 3, Be = 5;
  //각각 다항식 A, B의 계수, 차수를 적어놓은 데이터의 시작과 끝
  int avail = 6;
  //현재 위치
  Poly_Add(poly, As, Ae, Bs, Be, &avail);
  Print_Poly(poly, As, Ae, Bs, Be, avail);
}
```

[과제에 대한 고찰]

avail은 Print_Poly 함수로 다항식 3개를 출력할 때, 총 몇 항까지 있는지를 알기 위해서 Poly_Add 함수로 넘길 때 주소로 넘긴다. 그렇기 때문에 Poly_Add 함수에서 avail 안의 값을 1 증가시키거나 감소시킬 때는 포인터 변수를 이용해서 (*avail)++나 (*avail--)를 사용해야 한다. 그렇게 하지 않았을 때 실행시켜본 결과, avail의 주소값이 가감되었기 때문에 제대로 된 결과값이 출력되지 않았다.

또한, 교안의 의사코드에는 나오지 않았지만, 차수가 0인 항은 계수만 쓰는 상수항이기 때문에 Print_Poly 함수에 차수가 0인지 아닌지가 조건인 if-else 문을 추가했다.

다항식을 더하는 예제를 처음 사전실습했을 때, 배열에 값을 넣는 것과 반복문을 사용한 것까지는 비슷했다. 차이점을 꼽자면 반복문을 전체 차수에 맞춰서 돌렸고, (이번 실습을 코딩하면서) 이런 코드는 더해야 할 다항식의 정확한 최고차항의 차수를 알 수 없을 때 취약하다는 걸 알게 되었다. 그리고 이번에 실습한 이 코드는 내가 처음에 설계한 코드와 다르게 조금씩만 조정해주면 더하는 다항식의 수를 증가시켜도 쉽게 실행이 된다는 장점이 있다.