# Data Structure1 Report

순천향대학교


과목명 | 자료구조1

담당교수 | 홍 민

학과 | 컴퓨터소프트웨어공학과

학년 | 2학년

학번 | 20194059

이름 | 김태완

제출일 | 2020.06.18

# 목차

# 1. 두 다항식의 곱셈 연산을 수행하는 프로그램

## 1.1 문제 분석

SCH 순천향대학교

### 04 과제

- 연결 리스트를 이용하여 2개의 다항식을 곱하는 곱셈 연산을 구현하시오.
- 제출일: 2020년 6월 18일(목)
- data.txt 파일에는 아래와 같은 양식으로 데이터가 저장되어 있음

```
poly1 2 7 3 12 2 8 1 0
poly2 8 12 5 4 -3 10 10 6
```

이번 문제는 연결리스트를 이용하여 파일에서 읽어온 데이터를 활용해 두 다항식의 곱셈을 연산하는 프로그램을 작성 하는 문제이다. 이미 기존에 수업시간에 두 다항식의 덧셈을 연산하는 프로그램을 배웠기 때문에 파일로부터 데이터를 읽어오는 코드와, 덧셈 연산을 수행하는 코드를 곱셈 연산으로 조금 바꾸기만 하면 크게 어려울 것 같지 않은 문제이다. 덧셈과 달리 다항식의 곱셈은 이중 반복문을 이용하면 코드를 작성 할 수 있을 것 같다.

## 1.2 소스코드

```c
/*****************************************************************************
작성자: 20194059 김태완
작성일: 2020.06.10
프로그램명: 링크드 리스트를 이용하여 두 다항식의 곱셈 연산을 수행하는 프로그램
*****************************************************************************/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

//필요한 데이터의 원소들을 모아둔 구조체
typedef struct Elements
{
    int coef; //계수
    int expon; //지수
}Elements;
typedef struct ListNode
{
    Elements elements; //필요한 데이터 구조체
    struct ListNode *link; //링크필드
}ListNode;
//연결리스트 헤더
typedef struct ListType
{
    int size; //연결리스트의 크기
    ListNode *head; //헤드노드의 주소
    ListNode *tail; //마지막 노드의 주소
}ListType;

void error(char *message); //오류 출력함수
ListType* creat(); //연결리스트 헤더를 생성하는 함수
ListNode* insert_node(ListNode *pre,Elements value); //pre의 뒤에 새로운 노드 생성
ListNode* insert_head(ListNode *pre,Elements value); //맨 앞에 새로운 노드 생성
void insert_last(ListType* plist,Elements elements); //연결리스트의 노드를 뒤에 삽입하는 함수
void insert_with_sort(ListType *plist,Elements value); //정렬과 동시에 삽입 연산을 해주는 함수
void read_data(FILE *fp,ListType* plist); //파일로부터 데이터를 읽어오는 함수
void poly_mul(ListType* plist1,ListType* plist2,ListType* plist3); //다항식의 곱셈연산을 수행하는 함수
void poly_print(ListType* plist); //다항식을 출력하는 함수

int main()
{
    FILE *fp; //파일포인터
    ListType *list1,*list2,*list3; //리스트헤더를 저장하는 변수
    fp = fopen("data.txt","r"); //data.txt를 읽기 모드로 오픈
    if(fp == NULL) error("파일을 열 수 없습니다."); //파일 오픈 오류시 경고문 출력
    list1 = creat(); //list1 헤더 생성
    list2 = creat(); //list2 헤더 생성
    list3 = creat(); //list3 헤더 생성
    read_data(fp,list1); //list1에 데이터를 읽어 정렬 후 연결리스트에 저장
    read_data(fp,list2); //list1에 데이터를 읽어 정렬 후 연결리스트에 저장
    printf("polynomial A = ");
    poly_print(list1); //list1 출력
    printf("polynomial B = ");
    poly_print(list2); //list2 출력
    poly_mul(list1,list2,list3); //list1과list2의 다항식 곱셈 연산 수행
    printf("polynomial C = ");
    poly_print(list3); //다항식 곱셈 결과 출력
    free(list1); //list1 메모리 반납
    free(list2); //list2 메모리 반납
    free(list3); //list3 메모리 반납
    return 0;
}
//에러 출력 함수
void error(char *message)
{
    fprintf(stderr,"%s₩n",message); //매개변수를 받아와 stderr형식으로 출력
    exit(1); //프로그램 종료
}
```

```
70  //연결리스트 헤더를 생성하는 함수
71 ListType* creat()
72  {
73      ListType *plist = (ListType *)malloc(sizeof(ListType)); //메모리 동적 할당
74      plist->size = 0; //크기 0으로 초기화
75      plist->head = NULL; //헤드노드를 NULL로 초기화
76      plist->tail = NULL; //마지막 노드를 NULL로 초기화
77      return plist; //plist를 반환
78  }
89  //맨 앞에 새로운 노드 생성
90 ListNode* insert_head(ListNode *pre,Elements value)
91  {
92      ListNode *p = (ListNode *)malloc(sizeof(ListNode));
93      if (p==NULL) error("메모리 할당 에러"); //동적할당 오류시 오류문 출력
94      p->elements = value;
95      p->link = pre;
96      return p;
97  }
79  //pre의 뒤에 새로운 노드 생성
80 ListNode* insert_node(ListNode *pre,Elements value)
81  {
82      ListNode *p = (ListNode *)malloc(sizeof(ListNode));
83      if (p==NULL) error("메모리 할당 에러"); //동적할당 오류시 오류문 출력
84      p->elements = value;
85      p->link = pre->link;
86      pre->link = p;
87      return p;
88  }
98  //연결리스트의 마지막 노드 뒤에 새로운 노드를 생성 후 삽입하는 함수
99 void insert_last(ListType* plist,Elements elements)
100 {
101     ListNode* temp = (ListNode *)malloc(sizeof(ListNode)); //메모리 동적 할당
102     if (temp==NULL) error("메모리 할당 에러"); //동적할당 오류시 오류문 출력
103     temp->elements = elements;
104     temp->link = NULL;
105     if(plist->tail == NULL)
106     {
107         plist->head = temp;
108         plist->tail = temp;
109     }
110     else
111     {
112         plist->tail->link = temp;
113         plist->tail = temp;
114     }
115     plist->size++;
116 }
```

```
117  //정렬과 동시에 삽입 연산을 해주는 함수
118 void insert_with_sort(ListType *plist,Elements value)
119  {
120      ListNode *p = plist->head;
121      ListNode *temp;
122      int n=0;
123      if(plist->head==NULL) //연결리스트가 비어있을 경우
124      {
125          insert_last(plist,value); //헤드노드 생성
126          n++;
127      }
128      for(temp=plist->head;temp;temp=temp->link)
129      {
130          if(value.expon>temp->elements.expon) //만약 받아온 지수가 원래 저장된 값보다 크다면
131          {
132              if(temp==plist->head) //그 위치가 헤드노드일 경우
133              {
134                  plist->head = insert_head(temp,value); //새로운 헤드노드를 생성하고 헤드 포인터를 바꾼다
135              }
136              else //헤드노드가 아닐경우
137              {
138                  insert_node(p,value); //temp의 앞에 새로운 노드 생성
139              }
140              n++;
141          }
142          p = temp; //p는 temp의 전 링크를 가르키게 함
143      }
144      if(n==0) //가장 작을때
145      {
146          insert_last(plist,value); //맨 뒤에 새로운 노드 생성
147      }
148  }
149  //파일로부터 데이터를 읽어오는 함수
150 void read_data(FILE *fp,ListType* plist)
151  {
152      char tmp[20];
153      Elements value;
154      fscanf(fp,"%s",tmp); //파일 각 줄의 첫번째에 있는 문자열을 읽어온다
155      while(fscanf(fp,"%d %d",&value.coef,&value.expon) == 2) //fscanf는 반환값으로 읽어온 데이터의 수를 반환하기
156      {                                                       //계수와 지수를 같이 읽어와 반환값이 2일때만 반복하
157          insert_with_sort(plist,value);
158      }
159  }
160  //다항식의 곱셈 연산을 수행해주는 함수
161 void poly_mul(ListType* plist1,ListType* plist2,ListType* plist3)
162  {
163      int n=0;
164      ListNode *a = plist1->head;
165      ListNode *b = plist2->head;
166      ListNode *tmp;
167      Elements elements;
168      for(a = plist1->head;a;a=a->link)
169      {
170          for(b = plist2->head;b;b=b->link)
171          {
172              elements.coef = a->elements.coef * b->elements.coef; //계수끼리는 곱해준다
173              elements.expon = a->elements.expon + b->elements.expon; //지수끼리는 더해준다
174              n=0;
175              for(tmp = plist3->head;tmp;tmp=tmp->link)
176              {
177                  if(tmp->elements.expon==elements.expon) //만약 list3에 지수값이 같은 항이 존재하면
178                  {
179                      tmp->elements.coef += elements.coef; //기존에 저장된 계수에 새로운 계수를 더해준다
180                      n++; //새로운 노드에 삽입 할 필요 없다는 것을 알려준다
181                  }
182              }
183              if(n==0) //새로운 노드에 삽입해야 하는 경우
184              {
185                  insert_last(plist3,elements);
186              }
187          }
188      }
189  }
```

```c
//다항식을 출력해주는 함수
void poly_print(ListType* plist)
{
    ListNode* p = plist->head;
    for(p = plist->head;p;p=p->link)
    {
        printf("%d^%d",p->elements.coef,p->elements.expon);
        if(p->link!=NULL) //맨 마지막에는 +출력을 하면 안된다
        {
            printf(" + ");
        }
    }
    printf("₩n");
}
```

## 1.3 소스코드 분석

```c
/*****************************************************************************
작성자: 20194059 김태완
작성일: 2020.06.10
프로그램명: 링크드 리스트를 이용하여 두 다항식의 곱셈 연산을 수행하는 프로그램
*****************************************************************************/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

```

1. 프로그램에 관한 정보가 담긴 주석과 프로그램 작성에 필요한 헤더파일을 추가한다.

```c
//필요한 데이터의 원소들을 모아둔 구조체
typedef struct Elements
{
    int coef; //계수
    int expon; //지수
}Elements;
```

2. 연결리스트의 필요한 데이터들(계수, 지수)을 모아둔 구조체 Elements를 정의한다

```c
typedef struct ListNode
{
    Elements elements; //필요한 데이터 구조체
    struct ListNode *link; //링크필드
}ListNode;
```

3. 연결리스트의 노드 구조체 ListNode를 정의한다

```c
//연결리스트 헤더
typedef struct ListType
{
    int size; //연결리스트의 크기
    ListNode *head; //헤드노드의 주소
    ListNode *tail; //마지막 노드의 주소
}ListType;

```

4. 연결리스트의 크기, 헤드, 테일에 대한 정보를 담은 구조체 ListType을 정의한다.

```
31  void error(char *message); //오류 출력함수
32  ListType* creat( ); //연결리스트 헤더를 생성하는 함수
33  ListNode* insert_node(ListNode *pre,Elements value); //pre의 뒤에 새로운 노드 생성
34  ListNode* insert_head(ListNode *pre,Elements value); //맨 앞에 새로운 노드 생성
35  void insert_last(ListType* plist,Elements elements); //연결리스트의 노드를 뒤에 삽입하는 함수
36  void insert_with_sort(ListType *plist,Elements value); //정렬과 동시에 삽입 연산을 해주는 함수
37  void read_data(FILE *fp,ListType* plist); //파일로부터 데이터를 읽어오는 함수
38  void poly_mul(ListType* plist1,ListType* plist2,ListType* plist3); //다항식의 곱셈연산을 수행하는 함수
39  void poly_print(ListType* plist); //다항식을 출력하는 함수
40
```

5. 프로그램 작성에 사용된 함수들에 대한 원형 정의를 해준다.

```
41  int main( )
42  {
43      FILE *fp; //파일포인터
44      ListType *list1,*list2,*list3; //리스트헤더를 저장하는 변수
45      fp = fopen("data.txt","r"); //data.txt를 읽기 모드로 오픈
46      if(fp == NULL) error("파일을 열 수 없습니다."); //파일 오픈 오류시 경고문 출력
47      list1 = creat( ); //list1 헤더 생성
48      list2 = creat( ); //list2 헤더 생성
49      list3 = creat( ); //list3 헤더 생성
```

6. main함수를 열고 필요한 변수들을 선언 해 주고, 파일을 읽기 모드로 연 다음 list1,2,3를 만들어 준다.

```
50      read_data(fp,list1); //list1에 데이터를 읽어 정렬 후 연결리스트에 저장
51      read_data(fp,list2); //list1에 데이터를 읽어 정렬 후 연결리스트에 저장
52      printf("polynomial A = ");
53      poly_print(list1); //list1 출력
54      printf("polynomial B = ");
55      poly_print(list2); //list2 출력
56      poly_mul(list1,list2,list3); //list1과list2의 다항식 곱셈 연산 수행
57      printf("polynomial C = ");
58      poly_print(list3); //다항식 곱셈 결과 출력
59      free(list1); //list1 메모리 반납
60      free(list2); //list2 메모리 반납
61      free(list3); //list3 메모리 반납
62      return 0;
63  }
```

7. 데이터를 파일로부터 읽어와 정렬하면서 연결리스트에 삽입 해준 후 ploy1,2를 출력 해 준다. 이후 다항식의 곱셈 연산을 진행 해 주고 곱셈 결과를 출력 한 후 메모리를 반납하고 프로그램을 종료한다.

```
64  //에러 출력 함수
65  void error(char *message)
66  {
67      fprintf(stderr,"%s₩n",message); //매개변수를 받아와 stderr형식으로 출력
68      exit(1); //프로그램 종료
69  }
```

8. 에러를 출력하는 함수를 정의 해준다.

```
70  //연결리스트 헤더를 생성하는 함수
71  ListType* creat( )
72  {
73      ListType *plist = (ListType *)malloc(sizeof(ListType)); //메모리 동적 할당
74      plist->size = 0; //크기 0으로 초기화
75      plist->head = NULL; //헤드노드를 NULL로 초기화
76      plist->tail = NULL; //마지막 노드를 NULL로 초기화
77      return plist; //plist를 반환
78  }
```

9. 연결리스트 헤더를 생성하는 함수를 정의 해준다.

```
79  //pre의 뒤에 새로운 노드 생성
80 ListNode* insert_node(ListNode *pre,Elements value)
81  {
82      ListNode *p = (ListNode *)malloc(sizeof(ListNode));
83      if (p==NULL) error("메모리 할당 에러"); //동적할당 오류시 오류문 출력
84      p->elements = value;
85      p->link = pre->link;
86      pre->link = p;
87      return p;
88  }
```

10. pre노드 뒤에 새로운 노드를 생성하여 삽입 해주는 함수를 정의 해준다.

```
89  //맨 앞에 새로운 노드 생성
90 ListNode* insert_head(ListNode *pre,Elements value)
91  {
92      ListNode *p = (ListNode *)malloc(sizeof(ListNode));
93      if (p==NULL) error("메모리 할당 에러"); //동적할당 오류시 오류문 출력
94      p->elements = value;
95      p->link = pre;
96      return p;
97  }
```

11. 맨 앞에 새로운 노드를 생성해 주는 함수를 정의한다.

```
98  //연결리스트의 마지막 노드 뒤에 새로운 노드를 생성 후 삽입하는 함수
99 void insert_last(ListType* plist,Elements elements)
100 {
101     ListNode* temp = (ListNode *)malloc(sizeof(ListNode)); //메모리 동적 할당
102     if (temp==NULL) error("메모리 할당 에러"); //동적할당 오류시 오류문 출력
103     temp->elements = elements;
104     temp->link = NULL;
105     if(plist->tail == NULL)
106     {
107         plist->head = temp;
108         plist->tail = temp;
109     }
110     else
111     {
112         plist->tail->link = temp;
113         plist->tail = temp;
114     }
115     plist->size++;
116 }
```

12. 연결리스트의 맨 마지막에 새로운 노드를 생성하여 삽인 해 주는 함수를 정의한다.

```
117  //정렬과 동시에 삽입 연산을 해주는 함수
118 void insert_with_sort(ListType *plist,Elements value)
119 {
120     ListNode *p = plist->head;
121     ListNode *temp;
122     int n=0;
123     if(plist->head==NULL) //연결리스트가 비어있을 경우
124     {
125         insert_last(plist,value); //헤드노드 생성
126         n++;
127     }
128     for(temp=plist->head;temp;temp=temp->link)
129     {
130         if(value.expon>temp->elements.expon) //만약 받아온 지수가 원래 저장된 값보다 크다면
131         {
132             if(temp==plist->head) //그 위치가 헤드노드일 경우
133             {
134                 plist->head = insert_head(temp,value); //새로운 헤드노드를 생성하고 헤드 포인터를 바꾼다
135             }
136             else //헤드노드가 아닐경우
137             {
138                 insert_node(p,value); //temp의 앞에 새로운 노드 생성
139             }
140             n++;
141         }
142         p = temp; //p는 temp의 전 링크를 가르키게 함
143     }
144     if(n==0) //가장 작을때
145     {
146         insert_last(plist,value); //맨 뒤에 새로운 노드 생성
147     }
148 }
```

13. 매개변수로 받아온 데이터를 정렬과 동시에 연결리스트에 삽입 해 주는 함수이다.

이때 연결리스트가 비어있을 경우 헤드 노드를 생성 하여 주고
temp를 이용해 앞의 노드들과 비교하며 지급 들어온 데이터가 기존의 데이터보다 더 클 경우 앞으로 보내줘야 한다.

이때 맨 앞으로 들어가야 하는 경우가 생기는데 이때는 insert_head()함수를 사용하면서 head의 주소값을 새로 만들어진 노드의 주소롤 바꿔 주었다.

맨 앞이 아닐 경우 temp의 전 노드의 주소값을 갖고 있는 p와 insert_node() 함수를 이용해 temp의 앞에 삽입 하여 준다.

삽입될 숫자가 가장 작아 맨 마지막에 위치해야 하는 경우 insert_last()함수를 사용하여 맨 마지막에 삽입 되도록 해준다.

```
149  //파일로부터 데이터를 읽어오는 함수
150 void read_data(FILE *fp,ListType* plist)
151 {
152     char tmp[20];
153     Elements value;
154     fscanf(fp,"%s",tmp); //파일 각 줄의 첫번째에 있는 문자열을 읽어온다
155     while(fscanf(fp,"%d %d",&value.coef,&value.expon) == 2) //fscanf는 반환값으로 읽어온 데이터의 수를 반환하기
156     {                                                        //계수와 지수를 같이 읽어와 반환값이 2일때만 반복하
157         insert_with_sort(plist,value);
158     }
159 }
```

14. 파일로부터 데이터를 읽어와 insert_witf_sort()함수를 통해 정렬 하면서 연결리스트에 하나씩 삽입 하여 주는 함수이다.

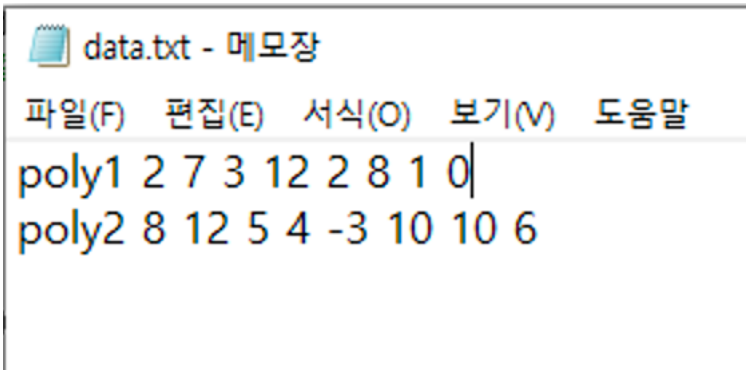
```
160  //다항식의 곱셈 연산을 수행해주는 함수
161 void poly_mul(ListType* plist1,ListType* plist2,ListType* plist3)
162 {
163     int n=0;
164     ListNode *a = plist1->head;
165     ListNode *b = plist2->head;
166     ListNode *tmp;
167     Elements elements;
168     for(a = plist1->head;a;a=a->link)
169     {
170         for(b = plist2->head;b;b=b->link)
171         {
172             elements.coef = a->elements.coef * b->elements.coef; //계수끼리는 곱해준다
173             elements.expon = a->elements.expon + b->elements.expon; //지수끼리는 더해준다
174             n=0;
175             for(tmp = plist3->head;tmp;tmp=tmp->link)
176             {
177                 if(tmp->elements.expon==elements.expon) //만약 list3에 지수값이 같은 항이 존재하면
178                 {
179                     tmp->elements.coef += elements.coef; //기존에 저장된 계수에 새로운 계수를 더해준다
180                     n++; //새로운 노드에 삽입 할 필요 없다는 것을 알려준다
181                 }
182             }
183             if(n==0) //새로운 노드에 삽입해야 하는 경우
184             {
185                 insert_last(plist3,elements);
186             }
187         }
188     }
189 }
```

15. 곱셈 연산을 수행하는 함수로 이중반복문을 통해 하나하나 곱해주지만 여기서 지수가 같은 항 끼리는 서로 더해 주어야 하기 때문에 반복문을 하나 더 돌려 같은 지수를 갖고 있는 항이 있는지 검사 한 이후 있다면 그 주소에 접근해 계수 끼리만 더해주어 연산 해준다

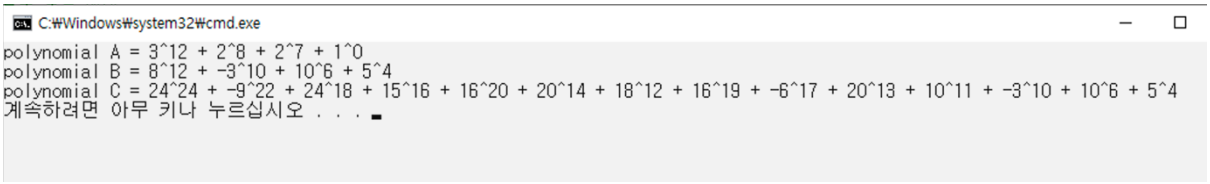
```
190  //다항식을 출력해주는 함수
191 void poly_print(ListType* plist)
192 {
193     ListNode* p = plist->head;
194     for(p = plist->head;p;p=p->link)
195     {
196         printf("%d^%d",p->elements.coef,p->elements.expon);
197         if(p->link!=NULL) //맨 마지막에는 *출력을 하면 안된다
198         {
199             printf(" + ");
200         }
201     }
202     printf("₩n");
203 }
```

16. 다항식을 출력하는 함수이다.

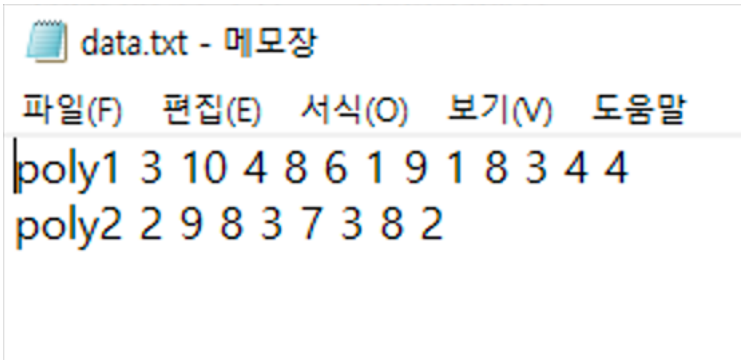
1.4 실행창



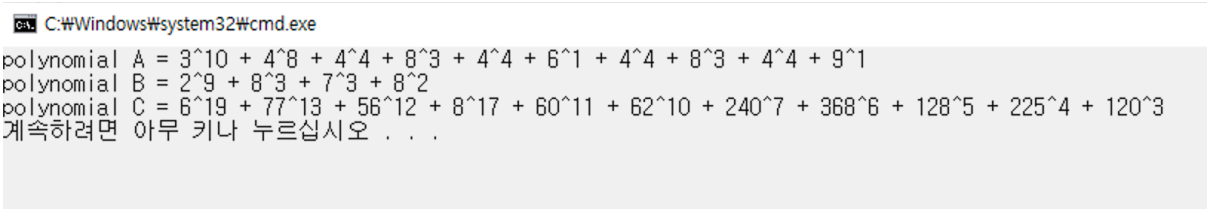

<data.txt -1>




<data.txt - 1 에 대한 출력>




<data.txt -2>




<data.txt - 2 에 대한 출력>

## 1.5 느낀점

처음 이 문제를 받았을 때는 그냥 배웠던 다항식의 덧셈에서 덧셈을 실행하는 함수만 곱셈으로 바꾸면 되는 정말 간단한 과제라고 생각 하였다. 하지만 데이터를 파일로부터 읽어오는 과정에서 정렬을 하면서 연결리스트에 삽입 하는게 너무나도 어려웠다. 진짜 열심히 코드를 작성하며 파일로부터 읽어오면서 poly1,poly2를 정렬하며 연결리스트에 삽입 하는 것 까지는 구현 할 수 있었지만 곱셈 결과를 정렬하며 연결리스트에 삽입하는 것은 끝내 하지 못했다. 굉장히 유사한 코드 일 것 같지만 곱셈 연산만 진행하면 계속 값이 중복으로 들어가 결국 시간이 부족해 그 오류를 찾아내지 못하고 정렬을 하지 못한 채로 제출 해야만 했다. 이번 2020학년도 1학기 과제는 이번이 마지막이지만 이번의 실패를 어머니로 삼아 다음 과제 부터는 조금 더 일찍 시작해서 꼭 마무리를 지을 수 있도록 노력 할 것이다.

# 3. 느낀점

벌써 2020학년도 1학기 자료구조 1 수업의 마지막 과제가 끝났다. 처음 첫번째, 두번째 과제를 할 때는 1학년때 c언어에서 배웠던 것들을 조금만 응용하면 쉽게 풀 수 있던 것들이라 어렵지 않게 풀었던 것 같은데, 세번째, 네번째 과제들은 자료구조 수업을 열심히 듣지 않으면 풀기 정말 어려운 과제들 이였다.
비록 이번 학기는 코로나 19로 인해 모든 수업을 온라인으로 수강 했지만 다음 학기때는 꼭 학교에서 수업을 들을 수 있기를 바란다.