REPORT

과 목 명 : 자료구조
담담교수 : 황두성 교수님
소　　속 : 소프트웨어학과
학　　번 : 32151671
이　　름 : 박민혁

단국대학교
Dankook University

1. 클래스를 이용한 구현

헤더파일

```
#pragma once

#include "Term.h"

class Polynomial
{
private:
    Term *termArray;
    int capacity;    // 배열의 크기
    int terms;       // 0이 아닌 항의 수
public:
    void Print(void);
    void NewTerm(const float theCoeff, const int theExp);
    Polynomial Add(Polynomial b);
    Polynomial Minus(Polynomial b);

    Polynomial(void);
    ~Polynomial(void);
};
```

```
#pragma once

#include "targetver.h"

#include <stdio.h>
#include <tchar.h>
```

```
#pragma once

#include <SDKDDKVer.h>
```

```
#pragma once

class Polynomial; //전방선언
class Term
{
friend Polynomial;
private:
    float coef;    // 계수
    int exp;       // 지수

public:
    Term(void);
    virtual ~Term(void);
};
```

소스 파일

```
#include "stdafx.h"
#include "Term.h"


Term::Term(void)
{
}


Term::~Term(void)
{
}


#include "stdafx.h"
```

```
#include "StdAfx.h"
#include "Polynomial.h"
#include "Term.h"

#include <iostream>
using namespace std;

Polynomial::Polynomial(void)
{
    capacity = 4;
    terms = 0 ;
    termArray = new Term[capacity] ;  // 크기 4 로 배열 생성
}

Polynomial::~Polynomial(void)
{

}

Polynomial Polynomial::Minus(Polynomial b)
{
    Polynomial c;
    int aPos = 0, bPos = 0;

    while ((aPos < terms) && (bPos < b.terms))
    {
        if (termArray[aPos].exp == b.termArray[bPos].exp)
        {
            float f = termArray[aPos].coef - b.termArray[bPos].coef;
            if (f) c.NewTerm(f, termArray[aPos].exp);
            aPos++;
            bPos++;
        }
        else if (termArray[aPos].exp < b.termArray[bPos].exp)
        {
            c.NewTerm(b.termArray[bPos].coef, b.termArray[bPos].exp);
            bPos++;
        }
        else
        {
            c.NewTerm(termArray[aPos].coef, termArray[aPos].exp);
            aPos++;
        }
```

```
    }

    for (; aPos < terms; aPos++)
        c.NewTerm(termArray[aPos].coef, termArray[aPos].exp);

    for (; bPos < b.terms; bPos++)
        c.NewTerm(b.termArray[bPos].coef, b.termArray[bPos].exp);

    return c;
}

Polynomial Polynomial::Add(Polynomial b)
// a(x)(*this의 값)와 b(x)를 더한 결과를 반환한다.
{
  Polynomial c;
  int aPos = 0, bPos = 0 ;

  while ((aPos < terms) && (bPos < b.terms))
    if(termArray[aPos].exp == b.termArray[bPos].exp) {
        float t = termArray[aPos].coef + b.termArray[bPos].coef;
        if (t) c.NewTerm(t, termArray[aPos].exp);
        aPos++; bPos++;
    }
    else if (termArray[aPos].exp < b.termArray[bPos].exp) {
        c.NewTerm(b.termArray[bPos].coef, b.termArray[bPos].exp);
        bPos++;
    }
    else {
        c.NewTerm(termArray[aPos].coef, termArray[aPos].exp);
        aPos++;
    }

  // A(x)(*this)의 나머지 항들을 추가한다.
  for (; aPos < terms; aPos++)
    c.NewTerm(termArray[aPos].coef, termArray[aPos].exp);

  // B(x)의 나머지 항들을 추가한다.
  for (; bPos < b.terms; bPos++)
    c.NewTerm(b.termArray[bPos].coef, b.termArray[bPos].exp);

  return c;
} // Add의 끝


void Polynomial::NewTerm(const float theCoeff, const int theExp)
// 새로운 항을 termArray 끝에 첨가한다.
{
    if (terms == capacity)
    {//termArray의 크기를 두 배로 확장
        capacity *= 2;
        Term *temp = new Term [capacity];          // 새로운 배열
        for(int i=0; i<terms ; i++)
            temp[i] = termArray[i] ;
        delete [ ] termArray;                      // 그전 메모리를 반환
        termArray = temp;
    }
    termArray[terms].coef = theCoeff;
    termArray[terms++].exp = theExp;
}


void Polynomial::Print( )
{
   int i ;

   cout << "₩n" ;
   if (terms) {
      for (i = 0 ; i < terms-1 ; i++)
          cout << termArray[i].coef << " x^" << termArray[i].exp << " + " ;
      // 마지막 항을 출력
      cout << termArray[i].coef << " x^" << termArray[i].exp << "₩n" ;
   }
   else
      cout << " No terms " ;

}
```

```cpp
#include "stdafx.h"


#include <iostream>
using namespace std;

#include "Polynomial.h"
#include "Term.h"

int _tmain(int argc, _TCHAR* argv[])
{
    Polynomial A, B, C;
    int i, n, e ;
    float c ;

    cout << "다항식 A의 항의 수 : " ;

    cin  >> n ;

    for ( i = 1; i <= n; i++ ) {
        cout << "다항식 A의 "<< i << " 번째 항의 계수와 지수 : " ;
        cin >> c >> e ;

        A.NewTerm(c, e) ;
    }

    cout << "다항식 B의 항의 수 : " ;
    cin >> n ;

    for ( i = 1; i <= n; i++ ) {
        cout << "다항식 B의 "<< i << " 번째 항의 계수와 지수 : " ;
        cin >> c >> e ;

        B.NewTerm(c, e) ;
    }

    C = A.Add(B) ;
    C.Print( ) ;

    C = A.Minus(B);
    C.Print( );


    return 0;
}
```

2. 연결리스트를 이용한 구현

헤더파일

```
#pragma once
#ifndef _CONTROLLER_H
#define _CONTROLLER_H
#endif
#include <iostream>
#include "node.h"
using namespace std;
class Controller
{
private:
    Node *root, *end, *poly;
public:
    Controller()
    {
        root = NULL; // 노드의 시작부분을 가리키게되는 포인터
        end = NULL; // 노드의 마지막부분을 가리키게되는 포인터
        poly = NULL; // 새로생성된 다항식노드를 가리키게되는 포인터

    }
    void setnode(float, int); // 노드생성 및 추가함수
    void insert(); // 다항식을 입력 받는 함수
    void display(); // 다항식을 출력 하는 함수
    void plus(Controller, Controller); // 두 다항식을 더하는 함수
    void multi(Controller, Controller); // 두 다항식을 곱하는 함수
    void result(float); // x의 값에 대한 결과를 연산 및 출력 하는 함수
};
```

```
#pragma once
#ifndef _NODE_H
#define _NODE_H
#endif
#include <iostream>
#include "poly.h"
using namespace std;
class Node // 노드는 자료를 저장하는 DATA부분과 다음 노드를 가리키는 Next 부분으로 이루어져 있다.
{
public:
    Poly *data; // 노드의 data부분 즉 다항식데이터가 들어있는 poly 클래스를 가리키는 포인터
    Node *next; // 다음 노드를 가리킬수있는 next 포이터
    Node()
    {
        next = NULL; // 초기는 NULL로 설정
        data = new Poly;
    }
};
```

```
#pragma once
#ifndef _POLY_H
#define _POLY_H
#endif
#include <iostream>
using namespace std;
class Poly
{
public:
    float coef; // 계수
    int exp; // 지수
    int term_count; // 항의 수
};
```

소스파일

```
#include "controller.h"
#include "node.h"
#include "poly.h"
void Controller::setnode(float coef, int exp)
{
    //////////* 첫 번째 노드일 경우 *//////////////
    if (root == NULL)
    {
        root = new Node; // 새 노드 생성
        end = root; // root 포인터와 end 포인터가 모두 첫 노드를 가리킨다.
        root->data->coef = coef; // 계수 저장
        root->data->exp = exp; // 지수 저장
    }
    else // 첫번째 노드가 존재할 경우
    {
        poly = new Node; // 노드를 추가한다
        end->next = poly; // 추가한 노드를 앞의 노드와 연결 시킨다
        end = poly; //end 포인터는 마지막 노드를 가리킨다.
        /* 새로 추가된 node에 데이터 저장 */
        poly->data->coef = coef; // 계수저장
        poly->data->exp = exp; // 지수저장
    }
}
void Controller::insert()
{
    int exp, before_exp = 1000; // 전번에 입력한 지수를 저장하기 위한 변수
    float coef = 0;
    while (1)
    {
        while (1) // 계수 입력 및 예외처리
        {
            cout << " 다항식 계수 입력 : ";
            cin >> coef;
            if (cin.fail()) // 에러 체크 함수
            {
                cout << " 숫자만 입력하세요 재 입력" << endl << endl;
                cin.clear();
                cin.ignore(256, '\n');
                continue;
            }
            else break;
        }
```

```
        while (1) // 지수 입력 예외처리
        {
            cout << " 다항식 지수 입력 : ";
            cin >> exp;
            if (cin.fail() || before_exp <= exp)
            {
            cout << " 숫자가 아니거나 이전의 지수와 값이 같거나 작습니다." << endl << endl;
            cin.clear();
            cin.ignore(256, '\n');
            continue;
            }
            else break;
        }
        if (coef != 0) setnode(coef, exp); // 노드를 생성 및 추가
        if (exp == 0) break;
        before_exp = exp;
    }
}
void Controller::display()
{
    poly = root;
    int count = 0; //항수를 count 합니다.
    while (1) // 첫노드 부터 마지막 노드까지 반복한다.
    {
        cout << poly->data->coef; // 계수 출력
        if (poly->data->exp == 0) cout << " "; // 지수가 0 일 경우
        else cout << "x" << "^" << poly->data->exp; // 지수가 0 이 아닐경우 문자 x와 지수값 출력
        count++;
        if (poly->next == NULL) break; // 마지막 노드라면 탈출
        cout << " + ";
        poly = poly->next; // 다음노드로
    }
    root->data->term_count = count; // 항수를 노드에 저장
}
void Controller::plus(Controller first, Controller second)
{
    first.poly = first.root;
    second.poly = second.root;
    float sum = 0;
    int first_terms = first.root->data->term_count; // 첫번째 다항식의 항수
    int second_terms = second.root->data->term_count; // 두번째 다항식의 항수
    while (first_terms != 0 && second_terms != 0) // 둘다 남은 항수가 존재 할경우 계속 반복, 즉 어느하나의
```

```
// 식이라도 모든 항의 체크가 끝나면 탈출
{
    if (first.poly->data->exp == second.poly->data->exp) // 두 다항식의 지수가 같다면
    {
        sum = first.poly->data->coef + second.poly->data->coef; // 두 항식을 더해라
        if (sum != 0) this->setnode(sum, first.poly->data->exp); // 세번째 노드에 첫번째 다항식과
        if (--first_terms > 0) // 체크할 항이 남아있다면
        {
            first.poly = first.poly->next; // 첫째 다항식의 다음 노드로
        }
        if (--second_terms > 0)// 체크할 항이 남아있다면
        {
            second.poly = second.poly->next;; // 둘째 다항식의 다음 노드로
        }
    }
    else if (first.poly->data->exp > second.poly->data->exp) // 첫번째 다항식의 지수 값이 크면
    {
        this->setnode(first.poly->data->coef, first.poly->data->exp); // 세번째 노드에 첫번째 다항식 값을 삽입
        if (--first_terms > 0)// 체크할 항이 남아있다면
        {
            first.poly = first.poly->next; // 첫번째 다항식을 다음노드로
        }
        else if (first.poly->data->exp < second.poly->data->exp) // 첫번째 다항식의 지수 값이 크면
        {
            this->setnode(second.poly->data->coef, second.poly->data->exp); // 세번째 노드에 두번째 다항식 값을 삽입
            if (--second_terms > 0)// 체크할 항이 남아있다면
            {
                second.poly = second.poly->next; // 두번째 다항식을 다음노드로
            }
        }
    }
    if (first_terms == 0) // 첫째 식에서 남은 항이 없으므로 두번째 남은 항을 내린다
    {
        while (second_terms)
        {
            this->setnode(second.poly->data->coef, second.poly->data->exp); // 세번째 노드에 두번째 다항식 값을 삽입
            if (--second_terms > 0)
            {
                second.poly = second.poly->next; // 두번째 다항식을 다음노드로
            }
        }
    }
        else if (second_terms == 0) // 둘째 식에서 남은 항이 없으므로 첫째 남은 항을 내린다
        {
            while (first_terms)
            {
                this->setnode(first.poly->data->coef, first.poly->data->exp); // 세번째 노드에 첫째 다항식 값을 삽입
                if (--first_terms > 0) // 체크할 항이 남아있다면
                {
                    first.poly = first.poly->next; // 첫째 다항식을 다음노드로
                }
            }
        }
    }
}
void Controller::multi(Controller first, Controller second)
{
    first.poly = first.root;
    second.poly = second.root;
    bool flag = true;
    float coef_multi = 0;
    int exp_sum = 0;
    int set_count = 0;
    int first_terms = first.root->data->term_count; // 첫째 항의 수
    int second_terms = second.root->data->term_count; // 둘째 항의 수
    while (first_terms != 0) // 첫째 식에 체크할 항이 남아있다면
    {
        while (second_terms != 0) // 둘째 식에 체크할 항이 남아있다면
        {
            coef_multi = first.poly->data->coef * second.poly->data->coef; // 계수 끼리 곱한다.
            exp_sum = first.poly->data->exp + second.poly->data->exp; // 지수 끼리 더한다
            this->poly = this->root;
            if (set_count > 0) // 항끼리의 연산결과가 처음 이 아니라면
            {
                while (1)
                {
                    if (exp_sum == this->poly->data->exp)
                        // 방금 계산해서 나온 지수끼리의 합과 결과 노드에 있는 지수가 같다면
                    {
                        this->poly->data->coef += coef_multi; // 결과 노드의 계수에 방금 나온 계수의 합을 더한다.
                        flag = false;
                    }
                    if (this->poly->next == NULL) break; // 다음노드가 없으면 탈출
                    this->poly = this->poly->next; // 결과 노드의 다음 노드로
```

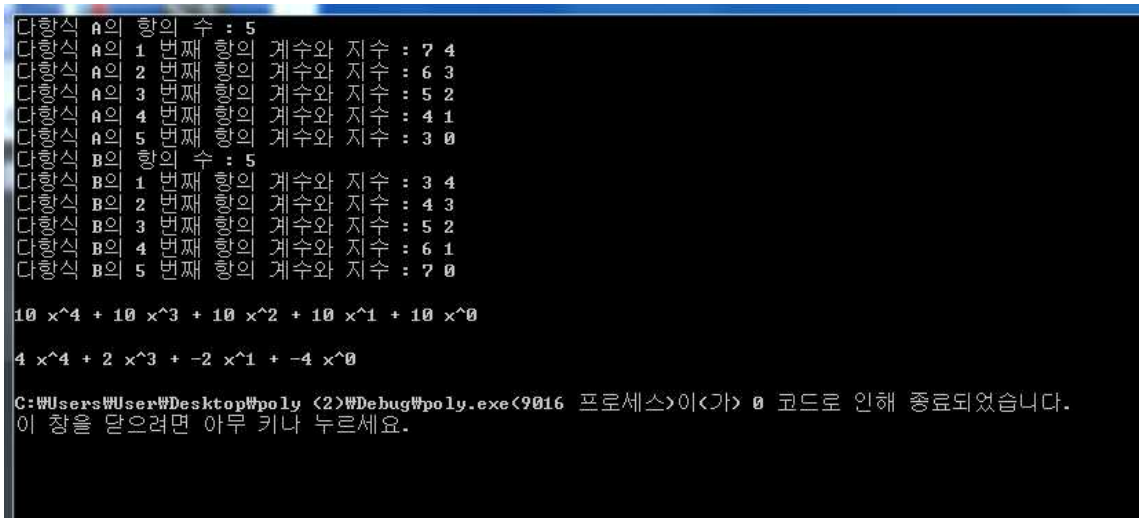
```
            }
        }
        if (flag) // 결과 노드의 지수들중 방금 연산한 지수의 합과 같은 값이 없다면
        {
            this->setnode(coef_multi, exp_sum); // 새로운 노드에 계수와 지수를 추가한다.
            set_count++;
        }
        flag = true;
        if (--second_terms > 0)
        {
            second.poly = second.poly->next; // 두번째 다항식을 다음노드로
        }
        // 두번째 다항식의 지수가 0 일때 까지 반복
    }
    second.poly = second.root; //두번째항 다시 반복하기위해 조건초기화
    second_terms = second.root->data->term_count; // 카운트도 초기화
    if (--first_terms > 0)
    {
        first.poly = first.poly->next; // 첫번째 다항식을 다음노드로
    }
}

}
void Controller::result(float x)
{
    poly = root;
    float result_value = 1;
    float total_value = 0;
    while (1)
    {
        for (int i = 0; i < poly->data->exp; i++) // x의 exp 승을 구한다.
        {
            if (poly->data->exp > 0) result_value *= x;
            else if (poly->data->exp < 0) result_value /= x;
        }
        result_value *= poly->data->coef; // 위의 결과 값에 계수를 곱한다.
        if (poly->next == NULL)
        {
            total_value += result_value; // 마지막 항를 더한다
            break; // 그리고 종료
        }
        else poly = poly->next;
```

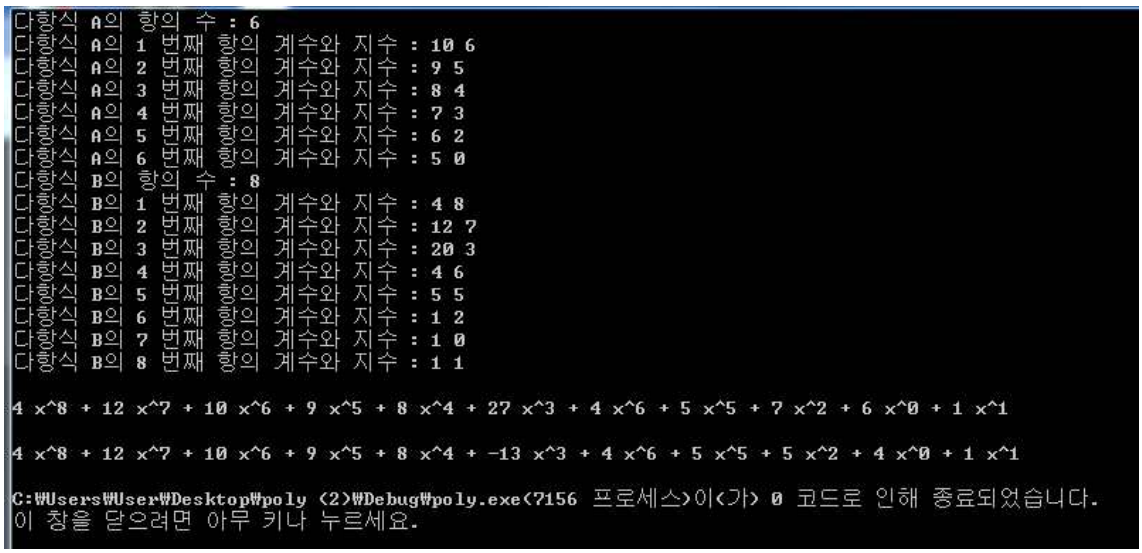
```
        else poly = poly->next;
        total_value += result_value; // 각 항의 값들을 더한다.
        result_value = 1;
    }
    cout << " x 가 " << x << " 일때 결과 값은 : " << total_value << endl;
}
```

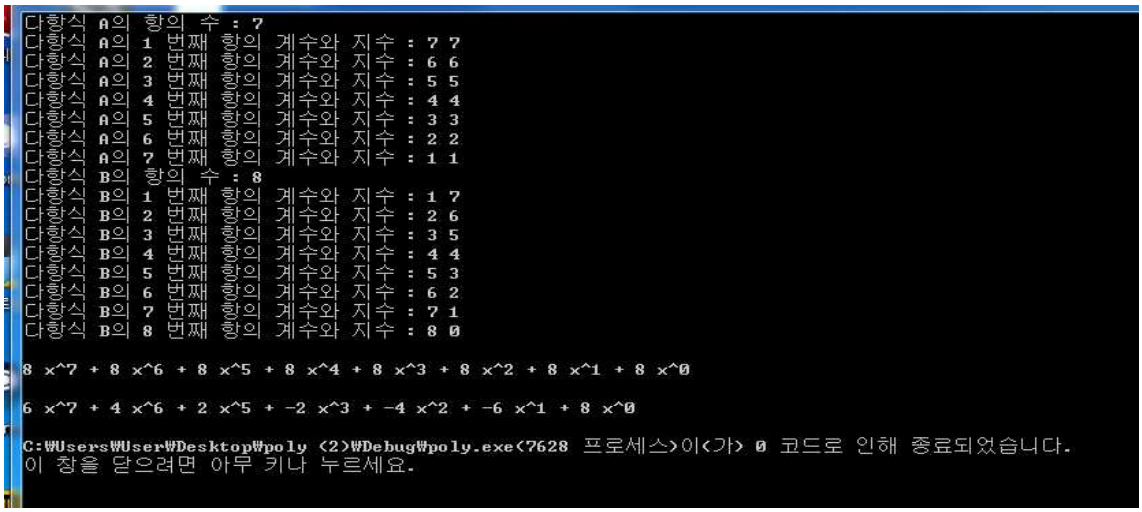
```
#include "controller.h"
int main(void)
{
    Controller control1; // 첫 번째 다항식을 위해
    Controller control2; // 두 번째 다항식을 위해
    Controller sum; // 덧셈 다항식을 위해
    Controller control3; // 세 번째 다항식을 위해
    Controller multi; // 곱셈 다항식을 위해
    float x = 0;
    cout << endl << "1번째 다항식" << endl << endl;
    control1.insert();
    cout << endl << "-------- 생성된 다항식 ----------" << endl;
    control1.display();
    cout << endl << endl << "2번째 다항식" << endl << endl;
    control2.insert();
    cout << endl << "-------- 생성된 다항식 ----------" << endl;
    control2.display();
    multi.multi(control1, control2);
    cout << endl << "-------- 곱의 결과 ----------" << endl;
    multi.display();
    cout << endl << endl << "3번째 다항식" << endl << endl;
    control3.insert();
    cout << endl << "-------- 생성된 다항식 ----------" << endl;
    control3.display();
    sum.plus(multi, control3);
    cout << endl << "-------- 합의 결과 ----------" << endl;
    sum.display();
    while (1)
    {
        cout << endl << endl;
        cout << "x의 값을 입력 하세요 : ";
        cin >> x;
        if (cin.fail())
        {
            cout << " 숫자만 입력하세요 재 입력" << endl << endl;
            cin.clear();
            cin.ignore(256, '\n');
            continue;
        }
        else break;
    }
    sum.result(x);
```

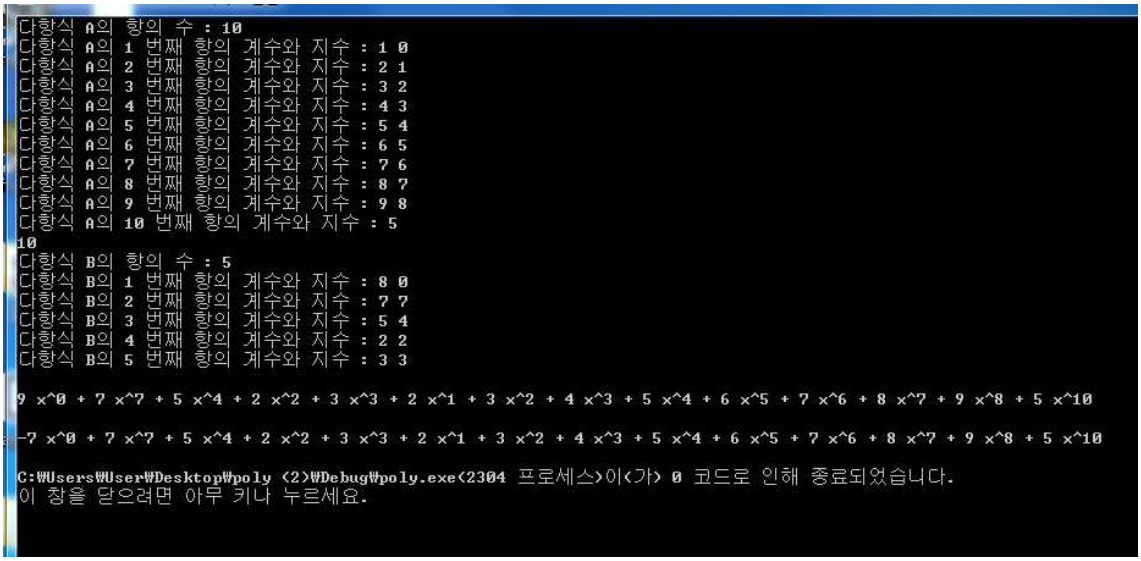
```
    sum.result(x);
    return 0;
}
```

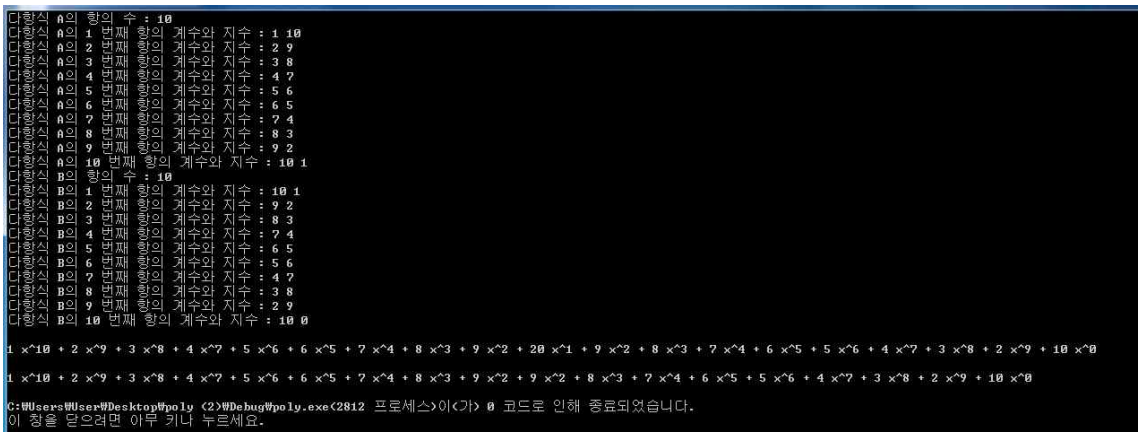
수행 결과


다항식 A의 항의 수 : 5
다항식 A의 1 번째 항의 계수와 지수 : 7 4
다항식 A의 2 번째 항의 계수와 지수 : 6 3
다항식 A의 3 번째 항의 계수와 지수 : 5 2
다항식 A의 4 번째 항의 계수와 지수 : 4 1
다항식 A의 5 번째 항의 계수와 지수 : 3 0
다항식 B의 항의 수 : 5
다항식 B의 1 번째 항의 계수와 지수 : 3 4
다항식 B의 2 번째 항의 계수와 지수 : 4 3
다항식 B의 3 번째 항의 계수와 지수 : 5 2
다항식 B의 4 번째 항의 계수와 지수 : 6 1
다항식 B의 5 번째 항의 계수와 지수 : 7 0
10 x^4 + 10 x^3 + 10 x^2 + 10 x^1 + 10 x^0
4 x^4 + 2 x^3 + -2 x^1 + -4 x^0
C:₩Users₩User₩Desktop₩poly (2)₩Debug₩poly.exe(9016 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.


다항식 A의 항의 수 : 6
다항식 A의 1 번째 항의 계수와 지수 : 10 6
다항식 A의 2 번째 항의 계수와 지수 : 9 5
다항식 A의 3 번째 항의 계수와 지수 : 8 4
다항식 A의 4 번째 항의 계수와 지수 : 7 3
다항식 A의 5 번째 항의 계수와 지수 : 6 2
다항식 A의 6 번째 항의 계수와 지수 : 5 0
다항식 B의 항의 수 : 8
다항식 B의 1 번째 항의 계수와 지수 : 4 8
다항식 B의 2 번째 항의 계수와 지수 : 12 7
다항식 B의 3 번째 항의 계수와 지수 : 20 3
다항식 B의 4 번째 항의 계수와 지수 : 4 6
다항식 B의 5 번째 항의 계수와 지수 : 5 5
다항식 B의 6 번째 항의 계수와 지수 : 1 2
다항식 B의 7 번째 항의 계수와 지수 : 1 0
다항식 B의 8 번째 항의 계수와 지수 : 1 1
4 x^8 + 12 x^7 + 10 x^6 + 9 x^5 + 8 x^4 + 27 x^3 + 4 x^6 + 5 x^5 + 7 x^2 + 6 x^0 + 1 x^1
4 x^8 + 12 x^7 + 10 x^6 + 9 x^5 + 8 x^4 + -13 x^3 + 4 x^6 + 5 x^5 + 5 x^2 + 4 x^0 + 1 x^1
C:₩Users₩User₩Desktop₩poly (2)₩Debug₩poly.exe(7156 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.


다항식 A의 항의 수 : 7
다항식 A의 1 번째 항의 계수와 지수 : 7 7
다항식 A의 2 번째 항의 계수와 지수 : 6 6
다항식 A의 3 번째 항의 계수와 지수 : 5 5
다항식 A의 4 번째 항의 계수와 지수 : 4 4
다항식 A의 5 번째 항의 계수와 지수 : 3 3
다항식 A의 6 번째 항의 계수와 지수 : 2 2
다항식 A의 7 번째 항의 계수와 지수 : 1 1
다항식 B의 항의 수 : 8
다항식 B의 1 번째 항의 계수와 지수 : 1 7
다항식 B의 2 번째 항의 계수와 지수 : 2 6
다항식 B의 3 번째 항의 계수와 지수 : 3 5
다항식 B의 4 번째 항의 계수와 지수 : 4 4
다항식 B의 5 번째 항의 계수와 지수 : 5 3
다항식 B의 6 번째 항의 계수와 지수 : 6 2
다항식 B의 7 번째 항의 계수와 지수 : 7 1
다항식 B의 8 번째 항의 계수와 지수 : 8 0
8 x^7 + 8 x^6 + 8 x^5 + 8 x^4 + 8 x^3 + 8 x^2 + 8 x^1 + 8 x^0
6 x^7 + 4 x^6 + 2 x^5 + -2 x^3 + -4 x^2 + -6 x^1 + 8 x^0
C:₩Users₩User₩Desktop₩poly (2)₩Debug₩poly.exe(7628 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.





make file

all : main
main : main.o functions.o
	gcc -o main main -o functions,o
main.o : main.c functions.h
	gcc -c main.c
functions.o : functions.c functions..h
	gcc -c functions.c