

**Review #02**

이름:\_\_\_\_\_ 학번:\_\_\_\_\_

(제출 시 .vs 폴더, debug 폴더를 지운 다음 .zip 으로 압축 후 제출)

1. 아래의 소스코드를 기반으로 출력 조건을 만족하도록 복소수의 +, - 연산을 지원하는 프로그램을 작성하시오([시작 코드]에서 명시된 내용을 수정하지 말 것).

[조건]

- +연산자, -연산자를 main() 함수의 코드와 같이 사용 가능하도록 작성 함
- main()함수의 중요로 지적된 사항의 숫자는 소수점 사항을 명확히 적용할 것(소수점 및 해당 사항이 다를 경우 감점)

[시작 코드]

```
#include <iostream>
using namespace std;

class Complex {
    double re, im;
public:
    Complex(double r, double i) { re = r; im = i; }
    Complex() { re = 0; im = 0; }

    // + 연산자 구현

    // - 연산자 구현

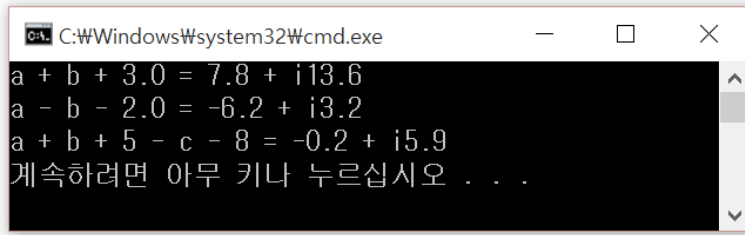
    void print(){
        cout << re << " + i" << im << endl;
    }
};

int main()
{
    Complex a(.3, 8.4), b(4.5, 5.2), c(2.0, 7.7);
    Complex sum, dif;
    sum = a + b + 3.0; // 중요 : "3.0"과 정확히 동일하게 기입하여야 함
    cout << "a + b + 3.0 = ";
    sum.print();

    dif = a - b - 2.0; // 중요 : "2.0"과 정확히 동일하게 기입하여야 함
    cout << "a - b - 2.0 = ";
    dif.print();

    sum = a + b + 5 - c - 8; // 중요 : "5, - 8"과 정확히 동일하게 기입하여야 함
    cout << "a + b + 5 - c - 8 = ";
    sum.print();
    return 0;
}
```

[실행결과]



```
C:\Windows\system32\cmd.exe
a + b + 3.0 = 7.8 + i13.6
a - b - 2.0 = -6.2 + i3.2
a + b + 5 - c - 8 = -0.2 + i5.9
계속하려면 아무 키나 누르십시오 . . .
```

2. 아래의 main()이 실행될 수 있도록 동적으로 할당된 array 들을 관리하는 class 를 구현 하시오.  
([시작 코드]에서 명시된 내용을 수정하지 말 것.)

[조건]

- VectorList 객체는 vItem 과 vLength 를 멤버 변수로 가지며, vItem 의 item 은 동적으로 할당된 배열의 pointer 를 저장하고, vLength 는 해당 item 의 배열 길이를 저장한다.
- VectorList 의 멤버 함수는 아래와 같다:
  - ~VectorList(): vItem 에 속한 모든 item(동적 배열의 포인터)을 할당 해제한다.
  - findVector(): 동적배열 포인터(\_pArray)와 배열의 길이(\_length)를 입력으로 받아 동일한 길이와 element(순서도 동일)를 가지는 배열이 vItem 에 존재하는지 확인하고, 존재 시 해당 배열의 인덱스를 반환, 그렇지 않으면 -1 을 반환한다.
  - print\_single\_vector()는 index 를 입력으로 받아 vItem 에서 해당 index 의 정보를 화면에 출력한다. 잘못된 인덱스(범위를 벗어난 인덱스)를 입력할 경우 "[Wrong index]"를 화면에 출력한다.
  - print\_all\_vectors()는 vItem 의 모든 정보를 화면에 출력하며, 이때 print\_single\_vector()를 함수 내부에서 사용한다.

[시작 코드]는 다음 페이지에 있음.

## [시작 코드]

```
#include <vector>
#include <iostream>
#include <iomanip>

using namespace std;

class VectorList {
    vector<double*> vItem; //Item (array)의 pointer저장
    vector<int> vLength; //각 Item(array)의 길이 저장
public:
    ~VectorList() { /*구현할 것*/ }
    void append(double* _pArray, int _length) { /*구현할 것*/ }
    int findVector(double* _pArray, int _length) { /*구현할 것*/ }
    void print_single_vector(int idx) { /*구현할 것*/ }
    void print_all_vectors() { /*구현할 것*/ }
};

int main() {
    VectorList vectorList;
    double* pTemp = new double[3]{ 1.1, 2.2, 3.3 };
    vectorList.append(pTemp, 3); // 아이템 추가

    pTemp = new double[5]{ 10.1, 10.2, 10.3, 10.4, 10.5 };
    vectorList.append(pTemp, 5); // 아이템 추가

    pTemp = new double[2]{ 4.5, 5.5 };
    vectorList.append(pTemp, 2); // 아이템 추가

    vectorList.print_single_vector(-1); // 1) -1번째 index의 vector를 화면에 출력
    vectorList.print_single_vector(0); // 2) 0번째 index의 vector를 화면에 출력
    cout << endl;

    vectorList.print_all_vectors(); // 3) 모든 vector들을 화면에 출력
    cout << endl;

    double* pTemp1 = new double[2]{ 4.5, 5.5 };
    double* pTemp2 = new double[4]{ 1.1, 2.2, 5.5, 4.4 };

    cout << "Index of vector{4.5, 5.5} : " << vectorList.findVector(pTemp1, 2) << endl;
    cout << "Index of vector{1.1, 2.2, 5.5, 4.4} : " << vectorList.findVector(pTemp2, 4)
    << endl;

    delete[] pTemp;
    delete[] pTemp1;
    delete[] pTemp2;

    return 0;
}
```

## [실행 화면]

```
1) [Wrong index]
2) 0-th vector:      1.1      2.2      3.3
3) 0-th vector:      1.1      2.2      3.3
   1-th vector:     10.1     10.2     10.3     10.4     10.5
   2-th vector:      4.5      5.5
Index of vector{4.5, 5.5} :2
Index of vector{ 1.1, 2.2, 5.5, 4.4 } :-1
```