Object Oriented Programming

Programming Assignment 2

Due: 2017/03/31

★Announcement★

1. 제출 기한은 3월 31일 자정까지입니다. LMS 시스템에서는 11시 58분 쯤부터는 업로드가 안 되니, 유의해주시기 바랍니다.
2. 과제는 늦게 제출하면 이유를 불문하고 0점입니다.
3. 채점은 Visual Studio 2015 환경에서 이루어집니다. 파일을 업로드하실 때, 작업하신 환경에 있는 **프로젝트가 들어있는 폴더를 그대로 압축**해서 올려주십시오. 그리고 폴더명은 문제1_학번의 형태로 만들어주십시오.
예시) `Problem1_20162939, Problem2_20162939, Problem3_20162939`
4. 문제마다 따로 프로젝트를 생성하여 따로 압축하여 제출하여 주시기 바랍니다. 즉, 총 **3개의 파일**을 제출하셔야 합니다. 위의 사안을 지키지 않거나 Visual Studio 2015 환경에서 컴파일이 되지 않을 경우 **상당한 감점**이 있습니다.
5. C++에서 배운 내용을 토대로 작성하셔야 합니다. C++로 구현 가능한 내용을 C로 구현했을 경우에는 감점으로 처리합니다.
예시) `scanf, printf, fprintf, fscanf, malloc, free`
6. 문서를 꼼꼼하게 읽고 문서에 나온 `method`들은 `input argument`와 `output result`를 문서에 맞게 작성해주세요.

1번 문제

다항식을 표현하는 `class Polynomial`을 구현한다. `Polynomial class`는 다음과 같은 member를 가진다.

- `int order` : 최고 차수
- `int *coefficients` : 각 계수를 상수항부터 차수가 낮은 순서대로 저장하는 배열. `order` 가 결정되면, 동적할당으로 배열을 생성한다.
  ex) $3x^2 + 4x + 2$는 `[2, 4, 3]`으로 저장된다.

`Polynomial class`는 다음과 같은 method를 가진다.

- `Polynomial()`
  - Default constructor로서 f(x)=0인 다항식을 생성한다.
- `Polynomial(int _order, int _coefficient[])`
  - 최고 차수가 `_order`이고, 각 계수가 `_coefficient`인 다항식을 생성한다. 각 계수는 상수항부터 차수가 낮은 순서대로 저장되어 있다.
  - `_coefficient`가 변할 때 object의 각 계수가 변하지 않도록 Copy 하는 것에 주의한다.
  - `_coefficient`의 array length와 `_order`가 서로 맞지 않는 상황은 고려하지 않는다.
- `~Polynomial()`
  - 사용한 동적으로 할당된 메모리를 모두 해제한다.
- `void add(Polynomial poly)`
  - 현재 다항식에 poly 다항식을 더하고 저장한다.
- `void sub(Polynomial poly)`
  - 현재 다항식에 poly 다항식을 빼고 저장한다.
- `void mul(Polynomial poly)`
  - 현재 다항식에 poly 다항식을 곱해 저장한다.
- `void div(Polynomial poly)`
  - 현재 다항식에 poly 다항식을 나눈 몫을 저장한다.
- `void mod(Polynomial poly)`
  - 현재 다항식을 poly 다항식으로 나눈 나머지를 저장한다.
- `void print()`
  - 현재 다항식을 출력한다.
  - 출력예) f(x) = 6x^2 + 4x + 3
- `int coefficient(int index)`
  - 차수가 index인 항의 계수를 리턴한다.
- `int substitute(int _x)`

- f(x)에 _x를 대입한 결과값을 리턴한다.

주어진 스켈레톤 코드(Problem2.zip)을 기반으로 작성하고, `main function`은 수정하지 말아야 한다. 반드시 위의 구현 사항을 만족하여야 하며, 추가적인 library의 사용은 허용하지 않는다.
추가적으로 필요한 method나 member가 있다면 구현에 추가해도 무방하다.
주어진 스켈레톤 코드의 파일명은 **절대** 수정하지 않는다. 수정할 경우 전체 점수에서 30% 감점.
채점은 추가적인 dataset을 이용할 수도 있다.

채점기준

1. `Accuracy` - 20 pts
2. `Program Structure` - 65 pts
   A. (7.5 pts) `add()` 구현
   B. (7.5 pts) `sub()` 구현
   C. (10 pts) `mul()` 구현
   D. (10 pts) `div()` 구현
   E. (10 pts) `mod()` 구현
   F. (3 pts) `print()` 구현
   G. (3 pts) `coefficient()` 구현
   H. (4 pts) `substitute()` 구현
   I. (10 pts) 생성자 및 소멸자 구현
3. `Code Readability` - 5 pts
4. `Comment` - 10 pts

2번 문제

아래와 같이 기본적인 벡터 연산을 하는 Vector 클래스를 구현하여, 클래스 Line, Plane을 정의하고 투영점을 구할 수 있는 라이브러리를 구현하여라.
(Vector 클래스는 Point를 표현하기 위해서도 사용될 수 있다.)

1. Vector: (x, y, z)

```
#define Point Vector

class Vector {
public:
	double x, y, z;

public:
	Vector();
	Vector(double x, double y, double z);
	static double innerProduct(Vector v1, Vector v2);
	static Vector outerProduct(Vector v1, Vector v2);
	static Vector plus(Vector v1, Vector v2);
	static Vector minus(Vector v1, Vector v2);
	static Vector multiply(double s, Vector v);
	double magnitude();
	Vector normalize();
	string toString();
};
```

- ✓ double innerProduct(Vector v1, Vector v2)
  - 두 벡터를 내적한 결과를 반환한다.
- ✓ Vector outerProduct(Vector v1, Vector v2)
  - 두 벡터를 외적한 결과를 반환한다.
- ✓ Vector plus(Vector v1, Vector v2)
  - 두 벡터의 합을 반환한다.
- ✓ Vector minus(Vector v1, Vector v2)
  - 두 벡터의 차를 반환한다.
- ✓ Vector multiply(double s, Vector v)
  - 벡터에 상수를 곱하여 반환한다.
- ✓ double magnitude()
  - 이 벡터의 크기를 반환한다.
- ✓ Vector normalize()
  - 이 벡터의 단위 벡터를 반환한다.

- ✓ `string toString()`
  - (x, y, z) 꼴의 string을 반환한다.

2. Line: $(\overrightarrow{P_1} - \overrightarrow{P_0})t + \overrightarrow{P_0} = \vec{l}t + \overrightarrow{l_0}$

```
class Line {
private:
	Vector l,l0;
public:
	Line(Point& po1, Point& po2);
	Point projectTo(Point& po);
	string toString();
};
```

- ✓ `Line(Point& po1, Point& po2)`
  - 두 점을 지나는 직선의 식을 구하여 l과 l0에 알맞은 값을 대입한다. 이때, l0의 값으로 po1를 대입한다.
- ✓ `Point projectTo(Point& po)`
  - 점 po를 이 직선에 투영한 점을 계산하여 반환한다.
- ✓ `string toString()`
  - `Line: (l.x, l.y, l.z)t + (l0.x, l0.y, l0.z)` 꼴의 string을 반환한다.

3. Plane: $\vec{n}(\vec{r} - \overrightarrow{r_0}) = \vec{0}$

```
class Plane {
private:
	Vector n, r0;
public:
	Plane(Point& po1, Point& po2, Point& po3);
	Point projectTo(Point& po);
	string toString();
};
```

- ✓ `Plane(Point& po1, Point& po2, Point& po3)`
  - 세 점을 지나는 직선의 식을 구하여 n과 n0에 알맞은 값을 대입한다. 평면이 결정되지 않는 경우는 고려하지 않는다. 이때, r0의 값으로 po1을, n의 값으로 평면의 단위 법선 벡터를 대입한다.
- ✓ `Point projectTo(Point& po)`
  - 점 po를 이 면에 투영한 점을 계산하여 반환한다.

- ✓ string toString()
  - Plane: (n.x, n.y, n.z)(r - (r0.x, r0.y, r0.z)) = 0 꼴의 string을 반환한다.

각 클래스마다 header file과 cpp file을 나누어서 구현하도록 한다. 추가적으로 필요한 method나 member가 있다면 구현에 추가해도 무방하다.
이 과제에서 main함수에 요구되는 구현 조건은 없다. 즉 main함수를 공백 함수로 두어도 되고, 다음과 같이 코드를 구성하여 구현한 class가 제대로 작동하는지 테스트해볼 수도 있다. 제출시 다른 클래스의 코드와 main.cpp코드를 포함한 프로젝트를 압축하여 제출한다.

>> main.cpp

```
#include <iostream>
#include "Vector.h"
#include "Line.h"
#include "Plane.h"

using namespace std;

int main()
{
	Point p1 = Point(0, 0, 0);
	Point p2 = Point(1, 1, 1);
	Line l1 = Line(p1, p2);
	cout << l1.toString() << endl;

	Point p3 = Point(1, 2, 1);
	Plane pl1 = Plane(p1, p2, p3);
	cout << pl1.toString() << endl;
}
```

채점 기준

1. Accuracy - 20 pts
2. Program Structure - 40 pts
   A. Vector Class 구현 20 pts
   B. Line Class 구현 10 pts
   C. Plane Class 구현 10 pts
3. Code Readability - 30 pts (각각 클래스마다 .h와 .cpp 파일로 나누지 않을 시 감점)
4. Comment - 10 pts

부록

Inner Product

$$\vec{\mathrm{u}} \cdot \vec{\mathrm{v}} = u_x v_x + \ u_y v_y + \ u_z v_z$$

Outer Product (cross product)

$$\vec{\mathrm{u}} \times \vec{\mathrm{v}} = \ \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} i - \ \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} j \ + \ \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} k$$

Normal Vector of Plane

$$\vec{\mathrm{n}} = \frac{\overrightarrow{(Q_2 - Q_1)} \times \overrightarrow{(Q_3 - Q_2)}}{|\overrightarrow{(Q_2 - Q_1)} \times \overrightarrow{(Q_3 - Q_2)}|}$$

Point Projection to Line

$$proj_{[\vec{\mathrm{l}}]}(\vec{\mathrm{P}}) = \ \frac{\vec{\mathrm{P}} \cdot \vec{\mathrm{l}}}{\vec{\mathrm{l}} \cdot \vec{\mathrm{l}}} \vec{\mathrm{l}}$$

Point Projection to Plane

$$\overrightarrow{\pi(\vec{\mathrm{P}})} = \vec{P} - ((\overrightarrow{P - Q_1}) \cdot \vec{n})\vec{n}$$

3번 문제

이번 학기에 개설되는 컴퓨터공학과 교과목은 다음과 같다.

학수번호	수업명	수업 시간	선수과목 학수번호
273	Digital System Design	Mon 11:00~12:15 Wed 11:00~12:15 Fri 14:00~16:30	
232	Object Oriented Programming	Mon 14:00~15:15 Wed 14:00~15:15	101
233	Data Structure	Tue 14:00~15:15 Thur 14:00~15:15	101
321	Programming Language	Tue 14:00~15:15 Thur 14:00~15:15	101
353	Computer Network	Mon 09:30~10:45 Wed 09:00~10:45	
515	Machine Learning	Mon 14:00~15:15 Wed 14:00~15:15	
507	Software Engineering	Mon 09:30~10:45 Wed 09:30~10:45	261 331 321
311	Computer Architecture	Mon 11:00~12:15 Wed 11:00~12:15 Wed 18:30~21:00	101 211 273
261	Discrete Math. for Computer Science	Tue 09:30~10:45 Thur 09:30~10:45	
442	Intro. to Artificial Intelligence	Mon 09:30~10:45 Wed 09:30~10:45	233
331	Algorithm	Tue 09:30~10:45 Thur 09:30~10:45	101 233
421	Database Systems	Tue 09:30~10:45 Thur 09:30~10:45 Tue 19:00~20:40	101 233
423	Design of Compiler	Mon 11:00~12:15 Wed 11:00~12:15 Thur 19:00~19:50	311 312
504	Advanced Operating System	Mon 11:00~12:15 Wed 11:00~12:15	312
514	Pattern Recognition	Tue 15:30~16:45 Thur 15:30~16:45	261
518	Linguistic Basis for Nau. Lan. Proc.	Tue 11:00~12:15 Thur 11:00~12:15	
702	Computer Security	Mon 14:00~15:15 Wed 14:00~15:15	312 311
703	Vision and Language	Mon 11:00~12:15	

		Wed 11:00~12:15	
700	Parallel Programming	Tue 11:00~12:15 Thur 11:00~12:15	312 311 423

이 목록의 각 행은 class CourseInfo라는 자료형으로 구현이 되며, 다음과 같은 형태로 구현되어 있다.

```
class CourseInfo{
public:
    CourseInfo(int AcademicNum, string classname, ClassTime* times, int*
prerequisite);
~CourseInfo();

int getAcademicNum();      //AcademicNum의 값을 반환한다.
   int* getprerequisite();//prerequisite의 값을 반환한다.
   ClassTime* gettimes(); //times의 값을 반환한다.
private:
   int AcademicNum;        //학수번호
   string classname;       //수업명
   ClassTime* times;       //수업 시간
   int* prerequisite;      //선수 과목 목록
};
```

- ✓ **CourseInfo(int AcademicNum, string classname, ClassTime* times, int* prerequisite);**
  - CourseInfo에 대한 생성자. AcademicNum 인자는 CourseInfo에 있는 AcademicNum에 할당하며, classname 인자는 CourseInfo에 있는 classname에 할당하며, times 인자는 CourseInfo에 있는 times에 할당하며, prerequisite 인자는 CourseInfo에 있는 prerequisite에 할당한다.
- ✓ **~CourseInfo()**
  - CourseInfo에 대한 소멸자. 동적 할당을 전부 해제한다.

여기서 새롭게 등장하는 Classtime에 대한 자료형은 다음과 같다.

```
class ClassTime{
public:
   ClassTime(Day classtime_day,int classtime_start,int classtime_end);
   ~ClassTime();

   Day getclasstime_day();       //classtime_day의 값을 반환한다.
   int getclasstime_start();     //classtime_start의 값을 반환한다.
   int getclasstime_end();       //classtime_end의 값을 반환한다.
private:
   Day classtime_day;            // 수업 날짜
   int classtime_start;   //수업 시작 시간
   int classtime_end;     //수업 종료 시간
};
```

참고로 자료형 Day는 enum Day = {Non, Mon, Thu, Wed, Thur, Fri, Sat, Sun} 으로 정의된다. 여기서 첫번째 Non은 아무 날짜도 아니라는 의미이며, 나머지는 각각 월화수목금토일을 의미하게 된다.

- ✓ **ClassTime(Day classtime_day,int classtime_start,int classtime_end);**
  - ClassTime 에 대한 생성자. Classtime_day 인자는 ClassTime 에 있는 classtime_day에 할당하며, classtime_start 인자는 ClassTime 에 있는 classtime_start에 할당하며, classtime_end 인자는 ClassTime 에 있는 classtime_end에 할당한다.
- ✓ **~ClassTime();**
  - ClassTime에 대한 소멸자. 동적 할당을 전부 해제한다.

이 때, 학생들이 모여서 컴퓨터공학과 수강 신청을 하려고 한다. 학생들에 대한 정보는 다음과 같다.

이름	학과	과거에 들은 과목들 학수번호
Choi	Math	101 233
Lim	Mech	101
Sim	CS	101 232 233 290 261 273 211 291
Cho	CS	101 233 273 211 321 331 353 312 332 341

학생들에 대한 자료형은 다음과 같다.

```
class StudentInfo {
public:
StudentInfo(string name, string department,int* PreCourses,int* Courses);
~StudentInfo();

bool Register(CourseInfo** courselist,int AcademicNum);
bool CheckPrerequisite(CourseInfo** courselist,int index);
void Drop(int AcademicNum);
string Print();

  string getname();       //name을 return한다.
  int* getcourses();      //courses를 return한다.
private:
  string name;                  //학생의 이름
  string department;      //학생이 속한 학과
  int* PreCourses;        //학생이 그 전에 들은 과목들
  int* courses;                 //학생이 이번에 들을 과목들

};
```

- ✓ **StudentInfo(string name, string department, int* PreCourses, int* Courses);**
  - StudentInfo 클래스에 대한 생성자이다. name 인자는 StudentInfo에 있는 name에 할당하며, department 인자는 StudentInfo에 있는 department에 할당하며, PreCourses 인자는 StudentInfo에 있는 PreCourses에 할당하며, Courses 인자는 StudentInfo에 있는 Courses에 할당한다. 다만, PreCourses는 컴퓨터공학과 학생들의 경우에는 sorting을 해야한다.(sorting의 경우에는 오름차순이나 내림차순이나 상관없다.)
- ✓ **~StudentInfo();**
  - StudentInfo 클래스에 대한 소멸자이다. 동적 할당이 된 변수들을 전부 해제한다.
- ✓ **bool Register(CourseInfo** courselist, int AcademicNum);**
  - courselist는 과목들에 대한 테이블을 나타내고, AcademicNum은 학수번호를 나타낸다. AcademicNum에 해당하는 과목을 courselist를 사용해서 신

청이 가능한지 확인하고, 가능하면 신청하도록 한다. 이 때, 신청이 불가능한 경우는 다음과 같다.

- 수업 시간이 겹친다.
- 과목을 수강하는 데 필요한 선수 과목들을 모두 들은 것은 아니다.

✓ **bool CheckPrerequisite(CourseInfo** courselist, int index);**

- index는 수강하고자 하는 과목이며, courselist를 통해서 선수 과목을 알아낸다. 그 다음에 각 선수 과목에 대해서 학생이 이미 들었는지 확인한다. 이 때, 컴퓨터공학과 학생들은 이미 들은 과목을 저장하는 PreCourses가 sorting 되어있으므로, binary search를 통해서 검사를 한다. 타학과 학생들은 그냥 linear search를 사용한다.

✓ **void Drop(int AcademicNum);**

- AcademicNum은 학수번호를 의미한다. 학수번호에 해당하는 과목을 courses에서 찾아 삭제한다.

✓ **string Print();**

- 첫번째 칸에 학생 이름, 그 다음 칸부터는 신청한 과목의 학수번호를 적어서 하나의 문자열로 만들어서 반환한다.
  - 예시) Lim 261 353 321 232

입력은 한 줄씩 받고, 크게 3가지의 유형이 있다.

1. 학생이 수강신청을 한다. 맨 앞에 1을 적고, 그 다음에는 학생 이름, 그 다음에는 수강하고자 하는 과목들을 적는다. 엔터가 등장하면 문장이 끝났다고 생각한다.
   A. 예) 1 Choi 311 321 515 507 232
2. 학생이 수강포기를 한다. 맨 앞에 2를 적고, 그 다음에는 학생 이름, 그 다음에는 포기하고자 하는 과목들을 적는다. 엔터가 등장하면 문장이 끝났다고 생각한다.
   A. 예) 2 Choi 311 321 515 507 232
3. 수강신청한 과목들을 출력한다. 모든 학생들이 수강신청한 과목들을 output.txt로 출력한다. 출력의 형태는 '이름 과목1 과목2 …'로 한다.(자세한 예시는 아래를 참고하면 된다.) 0을 입력하고 엔터를 받으면 된다.
   A. 예) 0

이 프로그램에서는 프로그램의 복잡성을 감안하여 아래의 왼쪽과 같은 input 하나만을 받는다. 그리고 출력은 오른쪽과 같이 나오며, output.txt를 출력한다. Input은 코드 내부에 문자열로 저장한 다음에 사용하도록 한다.(console 입력 금지, input.txt를 통한 입력 금지)

입력	출력
1 Choi 273 1 Lim 261 353 321 233 232	Choi 273 Lim 261 353 321 232

1 Sim 311 442 331 421 423 1 Cho 504 507 515 514 518 700 702 703 2 Sim 311 442 2 Cho 504 0 1 Cho 504 0	Sim 331 Cho 515 518 Cho 515 518 504

채점기준

1. Accuracy – 20 pts
2. Program Structure – 60 pts
   A. (20 pts) CourseInfo() 구현
      i. (10 pts) ClassTime 자료형 구현
      ii. (10 pts) 기타 함수 구현
   B. (40 pts) StudentInfo() 구현
      i. (10 pts) Register() 구현
      ii. (10 pts) CheckPrerequisite() 구현
      iii. (10 pts) Drop() 구현
      iv. (5 pts) Print() 구현
      v. (5 pts) 기타 함수 구현
3. Code Readability – 10 pts
4. Comment – 10 pts