Object Oriented Programming

Programming Assignment 4

Due: 2017/05/07

★Announcement★

1. 제출 기한은 5월 7일 자정까지 입니다. LMS 시스템에서는 제출자가 몰리면 업로드가 안 될 수 있으니, 유의해주시기 바랍니다.
2. 과제는 늦게 제출하면 이유를 불문하고 0점입니다.
3. 채점은 Visual Studio 2015 환경에서 이루어집니다. 파일을 업로드하실 때, 작업하신 환경에 있는 프로젝트 폴더를 그대로 압축해서 올려주십시오. 그리고 폴더 이름은 ProblemN_학번의 형태로 만들어주십시오. 프로젝트 폴더를 압축하지 않고 cpp 파일만 업로드하거나, Visual Studio 2015 환경에서 컴파일 되지 않을 경우 **큰 감점**이 있습니다.
   예시) `Problem1_20162939, Problem2_20162939`
4. C++에서 배운 내용을 토대로 작성하셔야 합니다. C++로 구현 가능한 내용을 C로 구현했을 경우에는 감점으로 처리합니다.
   예시) `scanf, printf, fprintf, fscanf, malloc, free`
5. 문서를 꼼꼼하게 읽고 문서에 나온 `method`들은 `input argument`와 `output result`를 문서에 맡게 작성해주세요.

1. Set Operation

Set class를 구현하여 집합 연산을 하는 프로그램을 구현하여라.

클래스 Set은 다음과 같이 구현을 충족시키도록 한다.

```
class Set {
private:
	int size;
	int* elements;
public:
	Set();
	Set(int size, int* elements);
	~Set();
	Set operator+(const Set& set);
	Set operator+(const int element);
	Set operator-(const Set& set);
	Set operator-(const int element);
	Set operator*(const Set& set);
	Set operator=(const Set& set);
	bool operator==(const Set& set);
	Set operator!();
	bool contains(const int element);
	bool contains(const Set& set);
	bool isEmpty();
	void clear();
	int getSize();
	string toString();
};
```

- ✓ **Set operator+(const Set& set);**
  합집합을 구한다.
- ✓ **Set operator+(const int element);**
  이 집합에 element를 추가한 집합을 구한다.
- ✓ **Set operator-(const Set& set);**
  차집합을 구한다.
- ✓ **Set operator-(const int element);**
  이 집합에 element를 뺀 집합을 구한다.
- ✓ **Set operator*(const Set& set);**
  교집합을 구한다.
- ✓ **Set operator=(const Set& set);**
  집합을 복사한다.
- ✓ **Set operator==(const Set& set);**
  집합이 같은지 비교한다.
- ✓ **Set operator!();**
  여집합을 구한다. 이 때, 전체 집합은 1부터 100까지 자연수의 집합이라고 가정한다.
- ✓ **bool contains(const Set& set);**
  set이 이 집합에 포함되는지 검사한다.

- ✓ **`bool contains(const int element);`**
  element가 이 집합에 포함되는지 검사한다.
- ✓ **`Set operator!(const Set& set);`**
  여집합을 구한다.
- ✓ **`bool isEmpty();`**
  이 집합이 공집합인지 검사한다.
- ✓ **`void clear();`**
  이 집합을 공집합으로 만든다.
- ✓ **`int getSize();`**
  size를 반환한다.
- ✓ **`string toString();`**
  집합의 elements를 "{e1, e2, e3, …}" 꼴로 출력한다.

main함수는 각 클래스마다 header file과 cpp file을 나누어서 구현하도록 한다.
이 과제는 main함수를 구현할 필요가 없다. 다음과 같은 코드로 프로그램을 테스트해볼 수 있다.

```
int main()
{
	int elements1[4] = {1,2,3,4};
	Set s1 = Set(4, elements);

	int elements2[3] = {3,4,5};
	Set s2 = Set(3, elements);

	Set s3 = s1 * s2;
	cout << s3.toString() << endl;
}
```

채점 기준

1. Accuracy - 50 pts
2. Program Structure - 10 pts
3. Code Readability - 10 pts
4. Comment - 10 pts

2. Vector and Matrix

클래스 Vector2, Matrix3x3, Link를 구현하고, 2D 공간에서 l1, l2, l3, theta1, theta2가 주어졌을 때, 다음과 같은 구조의 로봇 팔의 끝 점 P의 좌표를 계산하는 프로그램을 구현하여라.

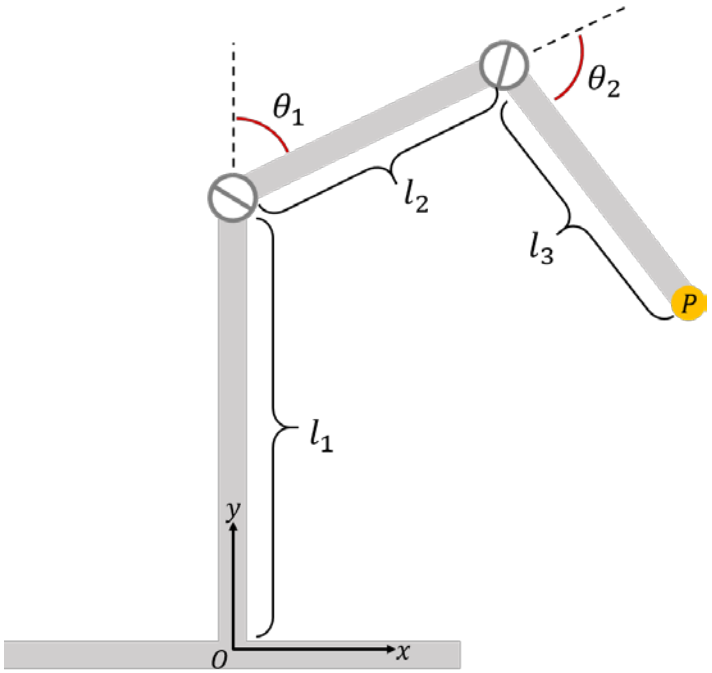

그림1

클래스 Vector2는 다음과 같이 구현을 충족시키도록 한다.

```
class Vector2 {
private:
	double x, y;
public:
	Vector2();
	Vector2(double x, double y);
	Vector2 operator+(const Vector2&);
	Vector2 operator-(const Vector2&);
	Vector2 operator*(double);
	friend Vector2 operator*(double, const Vector2&);
	Vector2 operator=(const Vector2&);
	string toString();
};
```

- ✓ **`Vector2 operator+(const Vector2&);`**
  두 벡터의 합을 계산한다.
- ✓ **`Vector2 operator-(const Vector2&);`**
  두 벡터의 차를 계산한다.
- ✓ **`Vector2 operator*(double);`**
  벡터의 상수 배를 계산한다.

- ✓ **friend Vector2 operator*(double, const Vector2&);**
  벡터의 상수 배를 계산한다.
- ✓ **Vector2 operator=(const Vector2&);**
  벡터를 복사한다.
- ✓ **string toString();**
  "(x,y)"꼴의 string을 반환한다.

한편, 행렬에 의한 이동, 회전, 크기 변환을 쉽게 하기 위해 동차 좌표계(Homogeneous Coordinate)를 이용한다. 동차 좌표계를 위한 행렬 클래스 Matrix3x3는 다음과 같이 구현한다.

동차좌표계에 대한 내용은 첨부 자료 "07.3Dtrans.pdf" (p4~p17)이나 "csed441_lecture7.pdf" (p41~p50)를 참고한다.

```
class Matrix3x3 {
private:
      double element[3][3];
public:
      Matrix3x3();
      Matrix3x3(double** element);
      Matrix3x3 operator+(const Matrix3x3&);
      Matrix3x3 operator-(const Matrix3x3&);
      Matrix3x3 operator*(const Matrix3x3&);
      Vector2 operator*(const Vector2&);
      Matrix3x3 operator*(const double);
      friend Matrix3x3 operator*(double, const Matrix3x3&);
      Matrix3x3 operator=(const Matrix3x3&);
      Matrix3x3 inverse();
      string toString();

      static Matrix3x3 identity();
      static Matrix3x3 translate(double x, double y);
      static Matrix3x3 rotate(double theta);
      static Matrix3x3 scale(double sx, double sy);
};
```

- ✓ **Matrix3x3 operator+(const Matrix3x3 &);**
  행렬의 합을 계산한다.
- ✓ **Matrix3x3 operator-(const Matrix3x3 &);**
  행렬의 차를 계산한다.
- ✓ **Matrix3x3 operator*(const Matrix3x3 &);**
  행렬의 곱을 계산한다.
- ✓ **Vector2 operator*(const Vector2 &);**
  행렬과 벡터의 곱을 계산한다.
- ✓ **friend Matrix3x3 operator*(double a, const Matrix3x3 &x);**
  행렬의 상수 배를 계산한다.
- ✓ **Matrix3x3 operator=(const Matrix3x3 &);**
  행렬을 복사한다.

- ✓ **`Matrix3x3 inverse();`**
  역행렬을 계산한다.
- ✓ **`string toString();`**
  "[e11, e12, e13; e21, e22, e23; e31, e32, e33;]" 꼴의 string을 반환한다.
- ✓ **`static Matrix3x3 identity();`**
  단위 행렬을 반환한다.
- ✓ **`static Matrix3x3 translate(double x, double y);`**
  x, y만큼 이동 변환하는 행렬을 반환한다.
- ✓ **`static Matrix3x3 rotate(double theta);`**
  theta만큼 회전 변환하는 행렬을 반환한다.
- ✓ **`static Matrix3x3 scale(double sx, double sy);`**
  x성분을 sx배, y성분을 sy배 시키는 행렬을 반환한다.

로봇 팔의 각 관절에 대한 클래스 Link는 다음과 같이 구현한다. Hierarchical Modeling에 관한 내용은 첨부 자료 "06.hierar.pdf" (p6~p31)를 참고한다.

```
class Link {
private:
      double length;
      double theta;
      Link* child;
public:
      Link(double l, double theta, Link* parent);
      Vector2 getEndPoint();
};
```

- ✓ **`Link* child;`**
  각 Link의 끝 점과 연결된 자식 Link를 말한다.
- ✓ **`Vector2 getEndPoint();`**
  Link의 끝 점의 좌표를 계산한다.

각 클래스마다 header file과 cpp file을 나누어서 구현하도록 한다. 추가적인 메서드나 멤버가 필요하다면 추가하여도 무방하다. 구현에 사용되지 않아도 제시된 메서드는 모두 구현하여야 한다.

main함수는 input.txt파일에 주어진 l1, l2, l3, theta1, theta2의 값을 읽고, 로봇 팔의 끝 점의 좌표를 계산하여 콘솔 창에 출력한다.

예제

input.txt	Console
1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 90.0 0.0 1.0 1.0 1.0 0.0 90.0	1. (0.0, 3.0) //input 번호 명시 2. (2.0, 1.0) 3. (1.0, 2.0)

input.txt는 l1, l2, l3, theta1, theta2순으로 저장하고 있고, Console 창은 각 입력에 대해 P의 위

치를 출력한다.

채점 기준

1. Accuracy - 60 pts
2. Program Structure - 40 pts
3. Code Readability - 10 pts
4. Comment - 10 pts