Homework #5

1. Rational Number class를 구현하시오.
이번 숙제는 연산자 오버로딩을 연습하는 것으로 유리수를 다루는 클래스를 만드는 것입니다. 주어진 파일은 아직 완성되지 않은 유리수 클래스입니다. 지금은 당연히 컴파일할 때 에러가 나겠지요? 여러분의 몫은 남겨진 부분들을 구현하여 제대로 돌아가게 하는 것입니다. 파일에 보면 주석으로 무엇을 해야 되는지 설명이 되어 있습니다.
클래스의 선언은 다음과 같습니다.

```
// 클래스 선언부

class RationalNumber {
private:
        int numerator;   // 분자
        int denominator;// 분모
        void reduction(); // 약분 (계산 후에는 반드시 약분을 해야한다.)

public:
        RationalNumber(int = 0, int = 1); // default 생성자
        // RationalNumber(int k); // k/1과 동일하게 생성

        RationalNumber operator + (const RationalNumber&); // 유리수와 유리수를 더함
        /* 이곳에 operator - 에 대한 프로토타입을 선언하세요. */
        RationalNumber operator * (const RationalNumber&);
        RationalNumber operator / (const RationalNumber&);

        bool operator > (const RationalNumber&) const; // 유리수의 크기를 비교
        bool operator < (const RationalNumber&) const;
        /* 이곳에 operator <= 에 대한 프로토타입을 선언하세요. */
        bool operator >= (const RationalNumber&) const;
        bool operator == (const RationalNumber&) const;
        bool operator != (const RationalNumber&) const;

        friend ostream &operator << (ostream&, const RationalNumber&);
};
```

다음은 테스트해 볼 수 있는 메인 함수입니다.

```
void main() {
	RationalNumber c(7, 3);
	RationalNumber d(3, 9);
	RationalNumber x;

	x = c + d;
	cout << c << " + " << d << " = " << x << endl;
	x = c - d;
	cout << c << " - " << d << " = " << x << endl;
	x = c * d;
	cout << c << " * " << d << " = " << x << endl;
	x = c / d;
	cout << c << " / " << d << " = " << x << endl;

	RationalNumber a(2, 5), b(3), y;
	y = a * c / d + b;
	cout << y;
	// 다른 operator들을 test할 수 있는 코드를 각자 작성

	return 0;
}
```

결과값:

```
	7/3 + 1/3 = 8/3
	7/3 - 1/3 = 2
	7/3 * 1/3 = 7/9
	7/3 / 1/3 = 7
	.....
	....
```

2. 위에서 작성한 RationalNumber class가 다음의 IntNumber class를 상속하도록 고치시오.

```
class IntNumber {
private:
        int k;

public:
        IntNumber(int k = 0); // default 생성자

        IntNumber operator + (const IntNumber&); // 정수와 정수를 더함
        /* 이곳에 operator - 에 대한 프로토타입을 선언하세요. */
        IntNumber operator * (const IntNuber&);
        IntNumber operator / (const IntNumber&);

        bool operator > (const IntNumber&) const; // 정수의 크기를 비교
        bool operator < (const IntNumber&) const;
        /* 이곳에 operator <= 에 대한 프로토타입을 선언하세요. */
        bool operator >= (const IntNumber&) const;
        bool operator == (const IntNumber&) const;
        bool operator != (const IntNumber&) const;

        friend ostream &operator << (ostream&, const IntNumber &);
};
```

이렇게 RationalNumber에서 IntNumber를 상속 받으면 RationalNumber의 멤버 변수인 numerator는 IntNumber의 멤버 변수인 k를 사용할 수 있다. 또한, RationalNumber class object가 사실상 정수인 경우, +, -, *, / 등의 연산에서 계산량을 줄일 수 있다. 이를 어떻게 구현할 수 있는지 설명하고 코드로 구현하시오.

3. 회사 내 직원의 정보와 월급 혹은 시급을 디스플레이해 주는 시스템을 만들고자 한다. 직원들은 정규직과 계약직으로 나누어져 있다. 정규직은 월마다 일정한 금액이 지급되고, 계약직은 시간을 단위로 금액이 지급된다고 하자. 위와 같은 class 구조는 다음과 같다.

직원(Employee):
멤버 변수로는 이름(문자열), 성(문자열), 주민등록번호(문자열), 직원 타입(int)이 있다. 직원 타입 0은 정규직, 1은 계약직이라고 하자. 멤버 함수로는 각각의 멤버 변수들의 설정자(set)와 접근자(get)가 있고, 직원의 이름과 나이, 주민등록번호, 고용, 채용일을 출력해주는 print() 가상 함수가 있다. 또한 double 반환 타입인 earning()이라는 순수 가상 함수가 존재한다.

● 설정자와 접근자의 이름은 set(멤버 변수), get(멤버 변수)로 한다.
ex) 이름에 대한 설정자와 접근자 : setName(), getName()

● 생성자는 매개변수로 이름, 성, 주민등록번호, 직원 타입

정규직(SalariedEmployee):
직원 클래스를 상속 받고, 멤버 변수로는 월급(double)이 있다. 멤버 함수로는 직원 클래스와 마찬가지로 멤버 변수인 월급게 대한 설정자(set)와 접근자(get)가 있고, 직원의 멤버 함수 print()를 오버라이딩하여 "₩n정규직원:₩n" 문구를 추가하여 직원의 정보들을 출력해주도록 해라. 직원의 순수 가상 함수인 earning()을 한달 월급을 반환하는 함수로 정의하자.

계약직(HourlyEmployee):
직원 클래스를 상속 받고, 멤버 변수로는 시간(double)과 시급(double)이 있다. 멤버 함수로는 두 멤버 변수의 설정자와 접근자가 있다. 시간은 최대 180시간까지만 인정된다고 하자(설정자에서 해당 조건 구현). 그 이상의 시간이 입력될 시에 0시간으로 초기화한다. 또 다른 멤버 함수로 직원의 멤버 변수 print()를 오버라이딩하여 "₩n계약직원:₩n" 문구를 추가하여 직원의 정보들을 출력해주도록 해라. 직원의 순수 가상 함수인 earning()을 시간에 따른 임금이 반환되도록 정의하자(시간 * 시급). 단 100시간을 초과하게 되면 초과된 시간에 시급이 1.5배가 해당 초과 시간에 맞춰 지급된다.

ex) 시간 = 165
100시간은 기본 시급, 나머지 65시간은 기본 시급 * 1.5

계약직원 배열과 정규직원 동적 배열을 만들어서 해당 객체들에 정보를 넣어주고 저장한 뒤에 정규직원 리스트와 계약직원 리스트를 모두 화면에 출력해주고, 해당 직원의 이름과 주민등록번호를 입력하면 해당 월급 혹은 임금이 출력될 수 있게 main()에서 구현해보자.

● 계약직원과 정규직원의 동적 배열 크기는 각각 5로 한다.

```
#include.. // 필요한 헤더 포함

int main() {
	// 정규직원 배열과 계약직원 동적 배열 할당 및 선언 및 데이터 초기화
	// 화면에 모든 직원들의 정보를 출력
	while(...) {
		// 성, 이름과 주민번호를 입력 받아 해당 직원의 임금 혹은 월급 출력
		// 해당 직원이 정규직인 경우, 연봉도 추가적으로 출력
		// 일치하는 객체가 없으면 해당 정보를 가진 직원이 없음을 알리는 내용 출력
		// "exit"을 입력 받으면 해당 while-loop 종료
	}
	// 동적 메모리 반환
}
```

정규직:

성	이름	주민번호	타입	월급
정	진우	901211-1234567	0	3,000,000
박	다름	950116-2345678	0	2,800,000
김	민정	890530-2456789	0	3,200,000
맹	지훈	850416-1789543	0	4,000,000
장	기열	881201-1043253	0	3,800,000

계약직:

성	이름	주민번호	타입	시간	시급
김	치우	901213-1234568	1	120	8,500
구	지현	960523-2942243	1	80	8,000
이	은지	970830-1343234	1	172	7,900
류	성태	890208-1032147	1	75	8,900
곽	다현	910219-2324429	1	140	8,400