C++ INTERMEDIATE

실습 과제

과제 1. 람다 표현식과 지역변수 캡쳐

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

int main()
{
    //vector<int> x { 1,2,3,4,5,6,7,8,9,10 };
    int x[10]{ 1,2,3,4,5,6,7,8,9,10 };

    int y[10];

    int v1 = 10;

    // x의 모든 요소에 v1를 더해서 y 에 넣어 주세요
    // 조건 1. x 와 y가 배열이 아닌 vector인 경우도 코드 변경없이 실행되게 해주세요..
    // 조건 2. 람다 표현식을 사용해 주세요.
    transform( 여기를 채워 보세요. );

    for (auto n : y)
        cout << n << endl; // 11,12,13,14,15,16,17,18,19,20 나와야 합니다.
}
```

과제 2. “perfect forwarding”을 사용한 싱글톤 템플릿 만들기 입니다

아래 코드는 싱글톤을 만들기 위해서 CRTP를 사용하고 있는 template 입니다.

```
template<typename Type> class Singleton
{
protected:
    Singleton() = default;
private:
    Singleton(const Singleton&) = delete;
    Singleton& operator=(const Singleton&) = delete;
public:
    static Type& getInstance()
    {
        static Type instance;
        return instance;
    }
};
class Mouse : public Singleton<Mouse>
{
public:
    Mouse() {}
};
int main()
{
    Mouse& c = Mouse::getInstance();
}
```

위 코드의 문제는 Singleton의 파생 클래스는 반드시 디폴트 생성자가 있어야 한다는 점입니다.

위 코드를 수정해서 아래 처럼 사용할 수 있게 만들어 보세요

```
class Cursor : public Singleton<Cursor>
{
public:
    Cursor(int x, int y) {}
};
int main()
{
    Cursor& c = Cursor::getInstance(1, 2);
}
```

과제 3. move 지원 클래스 만들기

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

class People
{
    string name;
    string addr;
    int    age;
public:
    People(string n, string add, int a) : name(n), addr(add), age(a) {}

    // 복사 생성자, 대입연산자, Move 생성자, Move 대입연산자를 만드세요..
    // 실행 여부를 확인하기 위해 logging 해주세요
};

int main()
{
    People p1{ "홍길동", "서울", 20 };
    People p2 = p1;       // copy
    People p3 = move(p1); // move
}
```

위 People 클래스에

복사 생성자, 대입연산자, Move 생성자, Move 대입연산자

를 만들어 보세요

과제 4. 버퍼 이동하기

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

int main()
{
    // (3)번 과제에서 만든 People을 사용하세요.

    // 1. People을 힙에 10개 생성해 보세요..
    //    People{"홍길동", "서울", 20} 으로 초기화 하세요..
    //    힌트) 메모리 할당과 생성자 호출을 분리 하세요

    // 2. 위에서 만드는 배열(버퍼)를 20개 키워 보세요..
    //     A. 20개 버퍼를 새로 할당하고
    //     B. 기존에 버퍼의 내용을 새로운 버퍼에 옮기세요.
    //        move 생성자에 예외가 없다면 move 생성자로,
    //        예외가 있다면 복사 생성자로 옮기세요
    //     C. 버퍼의 새로운 공간은 {"unknwon", "unknown", 0} 으로 초기화 하세요

    // 3. 새로운 버퍼의 새용을 출력하세요.


    // 4. 버퍼을 줄이지는 말고 버퍼 끝에 있는 3개 객체를 파괴(명시적 소멸자 호출)해
보세요.


}
```

과제 5. STL vector와 유사한 클래스 만들기.

```
// 아래 Point 에는 디폴트 생성자가 없습니다. 디폴트 생성자를 만들지 말고.
// 아래 상태에 과제를 진행해 주세요..
class Point
{
public:
    Point(int a, int b) {}
};
// 아래 main 함수가 실행되도록 Vector를 구현해야 합니다.
template<typename T> class Vector
{
    T* buff;
    int sz;   // size
    int capa; // capacity ( 메모리 사용량)
public:
};

int main()
{
    // 1. 아래 한줄이 실행되게 해주세요..
    //    Point 10개를 위한 버퍼가 만들어져야 하고, Point(1,1)로 초기화 되어야 합니다.
    Vector<Point> v(10, Point(1,1));

    // 2. 버퍼가 20개로 할당 되고 기존 10개는 이동(복사)하고,
    //    새로운 10개는 Point(0,0)으로 초기화되게 하세요
    v.resize(20, Point(0,0));

    // 3. 버퍼 크기를 다시 10개로 줄여 주세요. - 실제 메모리는 줄이지 말고, 객체만
파괴(소멸자 호출)해 주세요
    v.resize(10);

    cout << v.size() << endl; // 10
    cout << v.capacity() << endl; // 20

    // 4. initializer_list 추가하기.
    Vector<int> v2{ 1,2,3,4,5,6,7,8,9,10 };

    // 5. ranged for 지원 하기
    for (auto n : v2)
        cout << n << endl; // 1,2,3,4,5,6,7,8,9,10
}
```