12-1번 Dept.h 파일

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

class Dept {
	int size;
	int* scores;
public:
	Dept(int size) { this->size = size; scores = new int[size]; }
	Dept(const Dept& dept);
	~Dept();
	int getSize() { return size; }
	void read();
	bool isOver60(int index);
};
```

Dept.cpp파일

```
#include <iostream>
#include "Dept.h"
using namespace std;

Dept::Dept(const Dept& dept) {
	size = dept.size;
	scores = new int[dept.size];
	for (int i = 0; i < dept.size; i++) {
		scores[i] = dept.scores[i];
	}
}
Dept::~Dept() {
	delete[] scores;
}
void Dept::read() {
	cout << "10개 점수 입력>> ";
	for (int i = 0; i < size; i++)
	{
		cin >> scores[i];
	}
}
bool Dept::isOver60(int index) {
	if (scores[index] > 60) {
```

```
            return true;
        }
        else
            return false;
}
```

main.cpp 파일

```
#include <iostream>
#include "Dept.h"
using namespace std;

int countPass(Dept dept) {
	int count = 0;
	for (int i = 0; i < dept.getSize(); i++)
	{
		if (dept.isOver60(i))
			count++;
	}
	return count;
}
int main() {
	Dept com(10);
	com.read();
	int n = countPass(com);
	cout << "60점 이상은 " << n << "명";
}
```

12-3번 Dept.h 파일

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

class Dept {
	int size;
	int* scores;
public:
	Dept(int size) { this->size = size; scores = new int[size]; }
	~Dept();
	int getSize() const { return size; }
	void read();
	bool isOver60(int index) const;
};
```

Dept.cpp 파일

```
#include <iostream>
#include "Dept.h"
using namespace std;

Dept::~Dept() {
	delete[] scores;
}
void Dept::read() {
	cout << "10개 점수 입력>> ";
	for (int i = 0; i < size; i++)
	{
		cin >> scores[i];
	}
}
bool Dept::isOver60(int index) const {
	if (scores[index] > 60) {
		return true;
	}
	else
		return false;
}
```

main.cpp 파일

```
#include <iostream>
#include "Dept.h"
using namespace std;

int countPass(const Dept& dept) {
	int count = 0;
	for (int i = 0; i < dept.getSize(); i++)
	{
		if (dept.isOver60(i))
			count++;
	}
	return count;
}
int main() {
	Dept com(10);
	com.read();
	int n = countPass(com);
	cout << "60점 이상은 " << n << "명";
}
```

12-1번 문제

객체에 대한 복사를 위해서는 깊은 복사가 필요하다. 문제에서 학생들의 점수를 저장하고, 60점 이상의 학생들을 구별하기 위해서 read메서드와 isOver60메서드를 작성했다.

배열에 학생들의 점수를 저장하기 위해 read메서드에서는 반복문을 사용하여 scores배열에 저장했다.

60점 이상의 학생들을 판단하기 위해서 if문에서 배열의 인덱스를 이용해 판단한 후 bool값을 리턴하도록 작성했다.

복사생성자 생성에서 멤버변수의 변형을 막기위해 const로 선언하고, dept에 대한 변수 size를 복사하고 배열 scores로 dept에 대한 배열로 선언한 후 복사했다.

12-3번 문제

복사생성자를 제거하고 오류가 나지 않으려면 객체에 대한 참조로 코드를 수정해야 한다.

countPass에서 객체의 참조를 가져와야하는데, "&"를 사용해서 참조를 가져왔다.

이때, 객체의 참조를 가져오기 때문에 코드를 잘못 작성해서 원본 훼손의 여지가 있다. 멤버 함수를 const로 선언함으로써 원본 훼손을 방지할 수 있게 작성했다.