과제#3

내용: 교제 P275-276 의 **12번 문제 (1),(3)**를 프로그래밍하여 제출 하시오. **(1)과(3)**번만 수행.

12번

(1): main()의 실행 결과가 다음과 같이 되도록 Dept 클래스에 멤버들을 모두 구현하고, 전체 프로그램을 완성하라.

(3): Dept 클래스에 복사 생성자를 제거하라. 복사 생성자가 없는 상황에서도 실행 오류가 발생하지 않게 하려면 어느 부분을 수정하면 될까? 극히 일부분의 수정으로 해결된다. 코드를 수정해보라.

12번 문제의 코드 설명

Dept 클래스:

- size: 배열의 크기를 저장합니다.
- scores: 동적으로 할당된 정수 배열의 포인터입니다.
- 생성자: 입력된 크기만큼의 배열을 동적으로 할당합니다.
- 복사 생성자: 객체가 복사될 때 원본의 점수를 안전하게 복사합니다.
- 소멸자: 동적으로 할당된 메모리를 해제하여 메모리 누수를 방지합니다.
- read(): 사용자가 점수를 입력받아 scores 배열에 저장합니다.
- isOver60(): 주어진 인덱스의 점수가 60 이상인지 확인합니다.

countPass 함수:

- Dept 객체를 참조로 받아 60점 이상인 학생의 수를 카운트합니다.

main 함수:

- Dept 객체를 생성하고, 사용자로부터 점수를 입력받은 후, 60점 이상인 학생 수를 출력합니다.

(1)번 문제

- 문제 정의 :

주어진 문제는 학생들의 성적을 입력받고, 그 중 60점 이상인 학생의 수를 세는 프로그램을 구현한다. 클래스 Dept를 정의하여 성적을 저장하고 관리하며, 사용자로부터 성적을 입력받고, 60점 이상인 학생 수를 계산하여 출력하는 기능이 필요하다. 현재 프로그램은 복사 생성자와 소멸자, 그리고 정수를 입력받는 read()와 60점 이상인지를 판별하는 isOver60()이 구현되지 않았다. 이것들을 구현하여 복사 및 메모리 관리 문제를 해결해야 한다.

- 문제 해결 방법 :

객체 복사: countPass 함수에서 Dept 객체를 참조로 전달하기 위해 복사 생성자를 구현한다.

>>

```
Dept::Dept(const Dept& dept) {
        this->size = dept.size;
        this->scores = new int[dept.size];
        for (int i = 0; i < this->size; i++) {
                scores[i] = dept.scores[i];
        }
}
```

메모리 관리: 동적 할당된 메모리를 소멸자를 통해 적절하게 해제하여 메모리 누수를 방지한다.

>>

```
Dept::~Dept() {
        delete[] scores;
}
```

점수 입력 받기: 동적으로 할당된 배열 scores의 for문을 이용하여 점수를 입력받는다.

>>

```
void Dept::read() {
        cout << "10개 점수 입력>>";
        for (int i = 0; i < size; i++) {
                cin >> scores[i];
        }
}
```

정확한 성적 판별: isOver60 메서드를 사용하여 입력된 성적이 60점 이상인지 확인한다.1
>>

```
bool Dept::isOver60(int index) {
	if (this->scores[index] > 60) {
		return true;
	}
	else
	{
		return false;
	}
}
```

- 아이디어 평가 :

int n = countPass(com); 에서 복사 생성자가 호출된다. Dept::Dept(const Dept& dept) 에서 객체를 복사해서 프로그램을 실행하고 정상적으로 작동한다

- 문제를 해결한 키 아이디어 또는 알고리즘 설명 :

클래스 설계: Dept 클래스를 설계하여 성적을 저장하고, 메서드를 통해 성적을 입력받고 판별할 수 있도록 한다.

동적 메모리 할당으로 인한 메모리 누스 방지: 생성자에서 학생 수에 맞춰 메모리를 동적으로 할당하였기에 소멸자에서 이를 해제하여 메모리 관리를 철저히 했습니다.

복사 생성자: 복사 생성자가 생성되지 않으면, 같은 공간을 향하게 되어서 메모리 오류 생기게 된다. 메모리 오류가 생기지 않게 하려면 깊은 복사 생성자를 만들거나 해야한다. 위의 코드에서 "int n = countPass(com);" 부분이 복사 생성자가 호출되는 코드 부분이다.

성적 판별 로직: isOver60 메서드를 통해 각 학생의 성적을 판별하고, countPass 함수에서 이 정보를 바탕으로 60점 이상인 학생 수를 카운트합니다.

(3)번 문제

- 문제 정의 :

1. Dept 클래스에 복사 생성자를 제거할 것

2. 복사 생성자가 없는 상황에서도 실행 오류가 발생하지 않게 코드의 극히 일부분만 수정할 것

- 문제 해결 방법 :

복사 생성자를 완전히 제거하고 참조 전달로 countPass 함수의 매개변수를 const Dept&로 변경하여 객체를 복사하지 않고 참조로 전달합니다.

>>

```
int countPass(Dept& dept) {
	int count = 0;
	for (int i = 0; i < dept.getSize(); i++) {
		if (dept.isOver60(i)) count++;
	}
	return count;
}
```

- 아이디어 평가 :

int n = countPass(com); 에서 객체를 참조하여 실행했을 때 정상적으로 작동한다.

- 문제를 해결한 키 아이디어 또는 알고리즘 설명 :

복사 생성자를 제거한 상태에서도 프로그램이 정상적으로 작동하도록 하려면, countPass 함수에서 Dept 객체를 복사하지 않고 참조로 전달하면 된다.