문제 정의

1번에서 만든 코딩에다 복사 생성자를 사용하지 않고 10명의 점수를 입력받고 60점 이상을 통과시키는 코드를 구현해야 한다.

문제 해결 방법

객체를 값이 아닌 const 참조로 전달하는 방법 (변수가 참조를 통해 변경되지 않는 참조) >> 복사 생성자 구현 안해도 됨, 객체가 복사되지 않음.

```
int countPass(const Dept& dept) { // 60점 이상으로 통과하는 학생의 수를 카운트하기, 참조로 전달함으로써 복사도 방지한다
    int count = 0;
    for (int i = 0; i < dept.getSize(); i++) {
        if (dept.isOver60(i)) count++;
    }
    return count;
}
```

아이디어 평가

const 참조를 사용하면 함수가 인수를 변경하는 오류가 나지 않고, 복사본이 만들어지지 않는다. 그 결과 속도도 조금 더 빨라진다.

문제를 해결한 키 아이디어 또는 알고리즘

~Dept(){ delete[] scores; }로 scores 배열에 할당된 메모리 해제

```
~Dept() { // 소멸자
    delete[] scores;
```

countPass(const Dept& dept) >> Dept 객체를 참조로 받아 복사 대신 참조로 전달을 하

고 복사 생성자의 호출을 방지함.

```
int countPass(const Dept& dept) { // 60점 이상으로 통과하는 학생의 수를 카운트하기
    int count = 0;
    for (int i = 0; i < dept.getSize(); i++) {
        if (dept.isOver60(i)) count++;
    }
    return count;
```

또한, dept.getSize()와 dept.isOver60(i)의 객체 상태를 변경하지 않게 클래스 내의 멤버 함수 내에 const를 선언한다. 하지 않으면 오류가 난다.

```
    int getSize() const { return size; }
    void read() {
        cout << size << "명의 학생 점수를 입력하세요." << endl;
        for (int i = 0; i < size; i++) {
            cin >> scores[i];
        }
    }
    bool isOver60(int index) const{
        return scores[index] >= 60;
    }
};
```

이렇게 하면 복사를 방지하여 복사 생성자가 필요 없이 1번과 똑같은 동작을 하는 코드를 구현할 수 있다.