

문제(1)

```
#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);
};
```

클래스는 문제에서 주어지므로
클래스의 구현부를 작성해 주면
된다.

```

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

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

✓ Dept::~~Dept() {
  |   if (scores)
  |       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;
  |   return false;
  |
  | }

```

Dept::Dept(const Dept& dept)는 Dept의 복사생성자이다. this->size = size가 아닌 this->size = dept.size로 하여 깊은 복사를 해주어야 오류를 막을 수 있다. 그렇게 size를 복사한 다음 그 size만큼 scores에 공간을 할당해 주고 for (int i = 0; i < size; i++)에서 scores에 dept.scores를 하나씩 복사해준다. 그다음 소멸자에서는 scores를 힙에 반환한다. Dept::read에서는 cin으로 for문을 이용해 size만큼의 점수를 입력 받아 scores에 저장한다. Dept::isOver60(int index)는 countPass에서 i값을 0부터 dept의 size까지 차례대로 받아오기에 받는 scores[index] >= 60으로 scores의 값을 처음부터 끝까지 비교하여 맞으면 true값을 리턴하여 countPass의 count값이 올라갈 수 있게 한다.

```

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

```

```

#define _CRT_SECURE_NO_WARNING
#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";
}

```

countPass와 main함수는 문제에 주어져 있다

실행결과

```
10개 점수 입력>> 10 20 30 40 50 60 70 80 90 100
```

```
60점 이상은 5명
```

```
C:\Users\dbstj\source\repos\ConsoleApplication2\x64\Debug\ConsoleApplication2.exe(프로세스 16608)이(가) 0 코드(0x0)와 함께 종료되었습니다.
```

```
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
```

```
이 창을 닫으려면 아무 키나 누르세요...
```

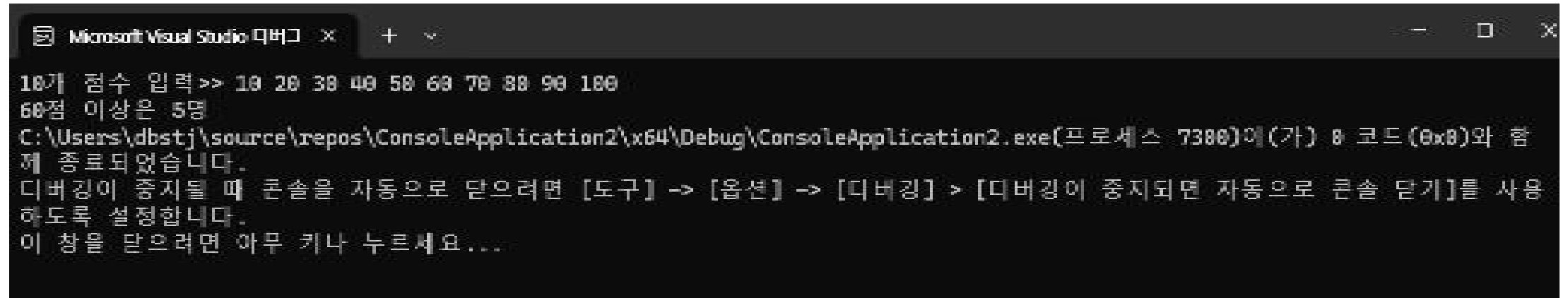
문제(3)

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

복사 생성자를 없애면 바꿔야 하는 곳은 복사 생성자가 실행 되는 곳이다. countPass를 보면 복사 생성자가 실행되는 걸로 알 수 있는데, 이 때 참조에 의한 호출을 사용하여 객체를 복사하지 않고 참조하여 코드를 진행할 수 있다.

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

실행결과



```
Microsoft Visual Studio 디버그 x + -  
10개 점수 입력>> 10 20 30 40 50 60 70 80 90 100  
60점 이상은 5명  
C:\Users\dbstj\source\repos\ConsoleApplication2\x64\Debug\ConsoleApplication2.exe(프로세스 7380)에(가) 0 코드(0x0)와 함  
께 종료되었습니다.  
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용  
하도록 설정합니다.  
이 창을 닫으려면 아무 키나 누르세요...
```