2번

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


template <class T> // 제네릭 함수를 만들기 위한 템플릿 선언 키워드
bool equalArrays(T data1[], T data2[], int n) { // 제네릭 타입 T형 매개변수 data1, data2
    for (int i = 0; i < n; i++) {
        if (data1[i] != data2[i]) { // 배열의 원소 비교해서 다르면,
            return false; // false 리턴
        }
        else { // 같으면,
            return true; // true 리턴
        }
    }
}
int main() {
    int x[] = {1, 10, 100, 5, 4}; // 예제 문제
    int y[] = {1, 10, 100, 5, 4}; // 예제 문제
    double k[] = { 2.3, 4.1, 4.1 }; // double형으로 해본 사례 추가
    double j[] = { 9.1, 3.2, 5.4 }; // k[]와 다르게 선언
    if (equalArrays(x, y, 5)) cout << "같다"; // 배열 x,y가 같으므로 "같다" 출력
    else cout << "다르다";
    cout << endl;
    if (equalArrays(k, j, 5)) cout << "같다"; // 배열 k,j가 다르므로 "다르다" 출력
    else cout << "다르다";
    cout << endl;
}
```

```
선택 C:₩WINDOWS₩system32₩cmd.exe
같다
다르다
Press any key to continue . . .
```

8번

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

class Comparable { // 추상 클래스
public:
    virtual bool operator > (Comparable& op2)=0; // 순수 가상 함수
    virtual bool operator < (Comparable& op2)=0; // 순수 가상 함수
    virtual bool operator == (Comparable& op2)=0; // 순수 가상 함수
};

class Circle : public Comparable { // 추상클래스를 상속받는 Circle
    int radius;
public:
    Circle(int radius=1) { this->radius = radius; } // 디폴트 radius값 = 1
    int getRadius() { return radius;}
    bool operator > (Comparable& op2) { //순수가상함수 구현함으로써 Circle은 온전한 클래스
                                        // 이때, virtual 속성은 상속되므로, 키워드 생략 가능
        // op2를 Circle* 으로 타입변환하여 -> 을 통해 멤버에 접근
        if (this->radius > ((Circle*)&op2)->getRadius()) { // this의 radius이 op2의 radius값보다 크면
            return true;
        }
        else {
            return false;
        }
    }
    bool operator < (Comparable& op2) { //순수가상함수 구현함으로써 Circle은 온전한 클래스
                                        // 이때, virtual 속성은 상속되므로, 키워드 생략 가능
        // op2를 Circle* 으로 타입변환하여 -> 을 통해 멤버에 접근
        if (this->radius < ((Circle*)&op2)->getRadius()) { // this의 radius이 op2의 radius값보다 작으면
            return true;
        }
        else {
            return false;
        }
    }
    bool operator == (Comparable& op2) { //순수가상함수 구현함으로써 Circle은 온전한 클래스
                                        // 이때, virtual 속성은 상속되므로, 키워드 생략 가능
        // op2를 Circle* 으로 타입변환하여 -> 을 통해 멤버에 접근
        if (this->radius == ((Circle*)&op2)->getRadius()) { //this의 radius이 op2의 radius값과 동일하면
            return true;
        }
        else {
            return false;
        }
    }
};

template <class T>
T bigger(T a, T b) { // 두 개의 매개 변수를 비교하여 큰 값을 리턴
    if (a > b) return a;
    else return b;
}

int main() { // 문제 예시부분
    int a=20, b=50, c;
    c = bigger(a, b);
    cout << "20과 50중 큰 값은 " << c << endl;

    Circle waffle(10), pizza(20), y; // waffle의 반지름이 10이고 pizza의 반지름이 20
    y = bigger(waffle, pizza); // 더 큰거를 y로 대입 // pizza의 반지름이 더 크므로 y에 pizza 대입
    cout << "waffle과 pizza 중 큰 것의 반지름은 " << y.getRadius() << endl; // y를 이용해 값 접근
    return 0;
}
```

```
C:₩WINDOWS₩system32₩cmd.exe
20과 50중 큰 값은 50
waffle과 pizza 중 큰 것의 반지름은 20
Press any key to continue . . .
```

14번

```cpp
#include <iostream>
#include <map> // map 헤더파일
using namespace std; // STL은 std 이름공간에 작성되었기에 using 지시어 필요

map<string, string> Program;  // 전역변수로 쓰여도 될 거 같아 이렇게 선언해봄
                              // 문자열 2개니까 <string,string> 으로 선언
                              // 맵 컨테이너 생성. key는 이름, value는 password
void insert() { // 삽입
    string Name, Password;
    cout << "이름 암호>> "; cin >> Name >> Password;
    Program.insert(make_pair(Name, Password)); // map의 insert 멤버 함수를 이용해
                                               // (Name, Password)를 pair로 저장
}

void examine() { // 검사
    string Name, Password;
    cout << "이름? "; cin >> Name;
    while (true) {
        cout << "암호? "; cin >> Password;
        if (Program.find(Name) == Program.end()) { // Key가 유효한지 확인
            cout << "Key를 찾을 수 없음" << endl;
        }
        else { // Key를 찾을 수 있다면,
            if (Program[Name] == Password) { // Key에 해당하는 Value가 Password와 동일하다면,
                cout << "통과!!" << endl;
                break; // 통과하였으면 break를 통해 while문 벗어남.
            }
            else {
                cout << "실패~~" << endl;
            }
        }
    }
}

int main() {
    int flag = 0; // while문을 벗어나기 위한 flag 변수
    cout << "***** 암호 관리 프로그램 WHO를 시작합니다 *****" << endl;
    while (true) {
        cout << "삽입:1, 검사:2, 종료3:>> ";
        int n; cin >> n;
        switch (n) {
        case 1:
            insert(); // n이 1이면 삽입
            break;
        case 2: // n이 2이면 검사
            examine();
            break;
        case 3: // n이 3이면 종료
            cout << "프로그램을 종료합니다..." << endl;
            flag = 1; // flag 변수를 1로 바꿔줌으로써 while문도 break할 수 있게 만듦
            break;
        default: // 1,2,3, 다 아니면,
            cout << "1,2,3 내에서 입력해주세요." << endl;
            break;
        }
        if (flag == 1) { // flag가 1이면 while문 break;
            break;
        }
    }
    return 0;
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
***** 암호 관리 프로그램 WHO를 시작합니다 *****
삽입:1, 검사:2, 종료3:>> 1
이름 암호>> Kim java
삽입:1, 검사:2, 종료3:>> 1
이름 암호>> lee C++
삽입:1, 검사:2, 종료3:>> 2
이름? lee
암호? C
실패~~
암호? 124
실패~~
암호? C++
통과!!
삽입:1, 검사:2, 종료3:>> 3
프로그램을 종료합니다...
Press any key to continue . . .
```