해밍 거리 (Hamming Distance)

해밍(Richard W. Hamming)은 1940년대 중반부터 컴퓨터/정보이론, 통신이론 연구분야에 참여하여 많은 기초이론을 개발한 수학자로서, 해밍이 처음 개발한 정보이론이나 통신이론에는 해밍의 이름을 붙인 용어들이 많이 존재한다.

어떤 이진수가 주어졌을 때, 이 이진수에 있는 1의 개수를 해밍 무게(Hamming weight)라고 한다. 예를 들어, 이진수 11101011011110011010001010 1 의 해밍 무게는 16이다. 또한, 두 이진수가 주어졌을 때, 두 이진수에서 같은 자리수 0 또는 1의 값이 서로 다른 자리수의 개수를 해밍 거리(Hamming distance)라고 한다.

예를 들어, 다음 두 이진수의 해밍 거리는 15이다. 아래 이진수의 밑줄 친 부분이 두 이진수에서 같은 자리수의 0 또는 1의 값이 서로 다른 경우를 나타낸다.

$$123456789_{(10)} = \underline{0}0\underline{0}1\underline{1}1\underline{0}1\underline{0}1\underline{1}1\underline{0}1\underline{1}1\underline{1}0\underline{0}1\underline{1}\underline{0}1\underline{0}\underline{1}0\underline{0}\underline{1}0\underline{1}0\underline{1}_{(2)}$$
$$987654321_{(10)} = \underline{1}1\underline{1}0\underline{1}0\underline{1}1\underline{0}1\underline{1}1\underline{1}0\underline{0}1\underline{1}0\underline{1}\underline{0}1\underline{0}\underline{0}\underline{0}1\underline{0}\underline{1}1\underline{0}\underline{0}\underline{0}1_{(2)}$$

두 개의 십진수가 주어졌을 때, 이 수를 이진수로 나타내었을 경우에, 이 두 이진수의 각각의 해밍 무게와 이 두 이진수 간의 해밍 거리를 계산하는 프로그램을 작성하시오.

입력

입력은 표준입력(standard input)을 사용한다. 입력은 t 개의 테스트 케이스로 주어진다. 입력 파일의 첫 번째 줄에 테스트 케이스의 개수를 나타내는 정수 t 가 주어진다. 두 번째 줄부터 t 개의 줄에는 한 줄에 한 개의 테스트 케이스에 해당하는 두 개의 정수 m n ($0 \leq m, n \leq 2^{31}-1$) 이 주어진다. 두 정수 사이에는 한 개의 공백이 있으며, 잘못된 데이터가 입력되는 경우는 없다.

출력

출력은 표준출력(standard output)을 사용한다. 입력되는 테스트 케이스의 순서대로 다음 줄에 이어서 각 테스트 케이스의 결과를 출력한다. 각 테스트 케이스에 해당하는 출력의 첫 줄에 입력되는 두 개의 십진수를 이진수로 표현하였을 때, 각 이진수의 해밍 무게를 입력되는 순서대로 출력하고, 또한 두 이진수 간의 해밍 거리를 출력한다. 출력되는 세 정수 사이에는 한 개의 공백을 둔다.

입력과 출력의 예

입력	출력
3 123456789 987654321 1 2 1 123456789	16 17 15 1 1 2 1 16 15

MyHammingDistance.h

```
#ifndef _MY_HAMMING_DISTANCE_H_
#define _MY_HAMMING_DISTANCE_H_

class MyBinaryNumber
{
public:
    // constructors
    MyBinaryNumber ();
    MyBinaryNumber (unsigned int val);

    // accessor/mutator functions
    unsigned int getValue() const;
    void setValue(unsigned int val);

    // utility function
    int getHammingWeight() const;
    int getHammingDistance(const MyBinaryNumber& bn) const;

private:
    unsigned int value;
};

#endif // _MY_HAMMING_DISTANCE_H_
```

MyHammingDistance.cpp

```
#include "MyHammingDistance.h"

#define XOR(a,b) (!(a)^!(b))

// constructors
MyBinaryNumber::MyBinaryNumber ()
:value(0)
{
}

MyBinaryNumber::MyBinaryNumber (unsigned int val)
:value(val)
{
}

// accessor functions
unsigned int MyBinaryNumber::getValue () const
{
    return value;
}

// mutator functions
void MyBinaryNumber::setValue (unsigned int val)
{
    value = val;
}

// 어떤 정수의 해밍 무게를 계산하는 함수
int MyBinaryNumber::getHammingWeight() const
{
}

// comptue Hamming Distance
int MyBinaryNumber::getHammingDistance(const MyBinaryNumber& bn) const

{
}
```

TestMyHammingDistance.cpp

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include "MyHammingDistance.h"
using namespace std;

int main()
{
    int numTestCases;

    cin >> numTestCases;

    for(int i=0; i<numTestCases; i++)
    {
        unsigned int num1, num2;

        cin >> num1 >> num2;

        MyBinaryNumber bn1(num1), bn2(num2);

        cout << bn1.getHammingWeight() << " " << bn2.getHammingWeight() << " "
             << bn1.getHammingDistance(bn2) << endl;
    }
    return 0;
}
```