IPv4 주소 변환

IPv4는 Internet Protocol version 4의 약어로서 인터넷 프로토콜(internet protocol)에서 주소를 표현하는 방법이다. IPv4의 주소는

xxx.xxx.xxx.xxx 123.234.111.222

와 같이 총 12자리 수로 표현되며, 12자리수는 문자 '.'(period)로 구분되는 세 자리수로 구성된 네 부분으로 나누어져 표현된다. 각 부분의 세 자리 정수는 0~255까지의 숫자를 나타낸다. 따라서, 각 부분은 8-비트, 즉 1-바이트로 나타낼 수 있고, IPv4 주소 전체는 32-비트로 나타낼 수 있다. 예를 들어, 아래의 예에서와 같이 IPv4의 주소에서 가장 왼쪽에 있는 부분을 32-비트 정수의 가장 높은 자리수에 해당하는 8-비트로 표현하며, 가장 오른쪽에 있는 부분을 32-비트 정수의 가장 낮은 자리수에 해당하는 8-비트로 표현하여, IPv4의 주소를 32-비트 정수로 나타낼 수 있다.

123.234.111.222

문자 '.'(period)로 구분된 IPv4 주소가 주어 졌을 때 이 주소를 위와 같은 방법으로 32-비트 정수로 변환하거나, 32-비트 정수가 주어졌을 때 이 정수를 IPv4 주소로 변환하는 프로그램을 작성하시오. 예를 들어, IPv4 주소 123.234.111.222는 정수 2078961630 로 변환된다.

입력

입력은 표준입력(standard input)을 사용한다. 입력은 t 개의 테스트 케이스로 주어진다. 입력의 첫 번째 줄에 테스트 케이스의 개수를 나타내는 정수 t 가 주어진다. 두 번째 줄부터 t 개의 줄에는 한 줄에 한 개의 테스트 케이스에 해당하는 데이터가 입력된다. 각 줄에서 첫 번째로 나타나는 데이터는 정수 k (k = 1, 2)로서, 정수 k 가 1인 경우는 같은 줄에 두 번째로 입력되는 데이터는 IPv4 주소를 나타내는 스트링이다. 정수 k 가 2인 경우는 같은 줄에 두 번째로 입력되는 데이터는 32-비트로 나타내어지는 정수 n ($0 \le n \le 2^{32}-1$)을 나타낸다. 두 데이터 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 데이터가 IPv4 주소인 경우에는 주소를 변환한 32-비트 정수를 십진수로 출력하고, 입력되는 데이터가 정수인 경우는 IPv4 주소로 변환한 스트링을 출력한다.

입력과 출력의 예

입력	출력
8 1 123.234.111.222 1 0.0.0.0 1 1.1.1.1 1 255.255.255.255 2 2078961630 2 0 2 16843009 2 4294967295	2078961630 0 16843009 4294967295 123.234.111.222 0.0.0.0 1.1.1.1 255.255.255.255

MyIpv4Address.h

```
#ifndef _MY_IPV4_ADDRESS_H_
#define _MY_IPV4_ADDRESS_H_

class MyIpv4Address
{
public:
    // constructors
    MyIpv4Address ();
    MyIpv4Address (unsigned int num);
    MyIpv4Address (char add[]);

    // utility functions
    void printAddress() const;
    void printNumber() const;

private:
    unsigned int addNumber;
    unsigned char address[4];

    void num2address();
    void address2num();
    void string2address(char add[]);
};

#endif // _MY_IPV4_ADDRESS_H_
```

MyIpv4Address.cpp

```
#include "MyIpv4Address.h"

// constructors
MyIpv4Address::MyIpv4Address ()
: addNumber(0)
{
        num2address();
}

MyIpv4Address::MyIpv4Address (unsigned int num)
: addNumber(num)
{
        num2address();
}

MyIpv4Address::MyIpv4Address (char add[])
{
        string2address(add);
        address2num();
}

// utility functions
void MyIpv4Address::printAddress() const
{
}

void MyIpv4Address::printNumber() const
```

```
{
}


// private functions
void MyIpv4Address::num2address()
{
}

void MyIpv4Address::address2num()
{
}

void MyIpv4Address::string2address(char add[])
{
}
```

TestMyIpv4Address.cpp

```
#include <iostream>
#include "MyIpv4Address.h"
using namespace std;

int main()
{
    int numTestCases;


    cin >> numTestCases;

    for (int i=0; i<numTestCases; i++)
    {
        unsigned int num;
        char add[16];
        int type;

        cin >> type;
        if (type == 1)
        {
            cin >> add;

            MyIpv4Address ipv4add(add);
            ipv4add.printNumber();
        }
        else if (type == 2)
        {
            cin >> num;
            MyIpv4Address ipv4add(num);
            ipv4add.printAddress();
        }
    }

    return 0;
}
```