CSE 111: Programming Language II

Problem 1) Happy Number

Source: http://online-judge.uva.es/p/v105/10591.html

Let the sum of the square of the digits of a positive integer S_0 be represented by S_1. In a similar way, let the sum of the squares of the digits of S_1 be represented by S_2 and so on. If $S_i = 1$ for some $i \geq 1$, then the original integer S_0 is said to be Happy number. A number, which is not happy, is called Unhappy number. For example 7 is a Happy number since 7 -> 49 -> 97 -> 130 -> 10 -> 1 and 4 is an Unhappy number since 4 -> 16 -> 37 -> 58 -> 89 -> 145 -> 42 -> 20 -> 4.

Input

The input consists of several test cases, the number of which you are given in the first line of the input. Each test case consists of one line containing a single positive integer **N** smaller than 10^9.

Output

For each test case, you must print one of the following messages:

```
Case #p: N is a Happy number.
Case #p: N is an Unhappy number.
```

Here **p** stands for the case number (starting from 1). You should print the first message if the number **N** is a happy number. Otherwise, print the second line.

Sample Input	Output for Sample Input
3 7 4 13	Case #1: 7 is a Happy number. Case #2: 4 is an Unhappy number. Case #3: 13 is a Happy number.

Problemsetter: Mohammed Shamsul Alam
International Islamic University Chittagong (IIUC)
Special thanks to Muhammad Abul Hasan

More Information:

- http://en.wikipedia.org/wiki/Happy_number
- http://mathworld.wolfram.com/HappyNumber.html

Problem 2) Kaprekar Number
Source: http://mathworld.wolfram.com/KaprekarNumber.html

Consider an n-digit number k. Square it and add the right $n$digits to the left $n$or $n-1$digits. If the resultant sum is k, then $k$is called a Kaprekar number. For example, 9 is a Kaprekar number since

$$9^2 = 81 \quad 8 + 1 = 9,$$

and 297 is a Kaprekar number since

$$297^2 = 88\,209 \quad 88 + 209 = 297.$$

The first few are 1, 9, 45, 55, 99, 297, 703, ...

Determine if a given number is **Kaprekar Number**

You can test your solution from http://online-judge.uva.es/p/v9/974.html

Problem 3) Odious Number
http://mathworld.wolfram.com/OdiousNumber.html

In odious number is a nonnegative number that has an odd number of 1s in its binary expansion. The first few odious numbers are therefore 1, 2, 4, 7, 8, 11, 13, 14, 16, 19, ... (Sloane's A000069). Numbers that are not odious are said to be evil numbers.

Determine if a given number is Odious or Evil number.

Problem 4) Narcissistic Number
http://mathworld.wolfram.com/NarcissisticNumber.html

Check if a number is Narcissistic Number or not.

An n-digit number that is the sum of the nth powers of its digits is called an n-narcissistic number. It is also sometimes known as an Armstrong number, perfect digital invariant (Madachy 1979), or plus perfect number. Hardy (1993) wrote, "There are just four numbers, after unity, which are the sums of the cubes of their digits:

$153 = 1^3 + 5^3 + 3^3$, $370 = 3^3 + 7^3 + 0^3$, $371 = 3^3 + 7^3 + 1^3$, and $407 = 4^3 + 0^3 + 7^3$. These are odd facts, very suitable for puzzle columns and likely to amuse amateurs, but there is nothing in them which appeals to the mathematician." Narcissistic numbers therefore generalize these "unappealing" numbers to other powers (Madachy 1979, p. 164).

The smallest example of a narcissistic number other than the trivial 1-digit numbers is

$$153 = 1^3 + 5^3 + 3^3. \tag{1}$$

The first few are given by 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474, 54748, ... (Sloane's A005188).

n	base-10 n-narcissistic numbers
1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
3	153, 370, 371, 407
4	1634, 8208, 9474

5 54748, 92727, 93084
6 548834
7 1741725, 4210818, 9800817, 9926315
8 24678050, 24678051, 88593477

Problem 5) The 3n + 1 problem
http://online-judge.uva.es/p/v1/100.html

The Problem

Consider the following algorithm:

1. input *n*
2. print *n*
3. if *n* = 1 then STOP
4. if *n* is odd then $n \longleftarrow 3n+1$
5. else $n \longleftarrow n/2$
6. GOTO 2

Given the input 22, the following sequence of numbers will be printed 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers *n* such that 0 < *n* < 1,000,000 (and, in fact, for many more numbers than this.)

Given an input *n*, it is possible to determine the number of numbers printed (including the 1). For a given *n* this is called the *cycle-length* of *n*. In the example above, the cycle length of 22 is 16. For any two numbers *i* and *j* you are to determine the maximum cycle length over all numbers between *i* and *j*.

The Input

The input will consist of a series of pairs of integers *i* and *j*, one pair of integers per line. All integers will be less than 1,000,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including *i* and *j*.

You can assume that no operation overflows a 32-bit integer.

The Output

For each pair of input integers *i* and *j* you should output *i*, *j*, and the maximum cycle length for integers between and including *i* and *j*. These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers *i* and *j* must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

Sample Input

```
1 10
100 200
201 210
900 1000
```

Sample Output

```
1 10 20
100 200 125
201 210 89
900 1000 174
```

Problem 6) Your Name in ASCII Art

Using one 1-dimensional array of Strings or one 2-dimensional **character** array to print one part of your name having at least 5 characters.

```
   ###    ##    ## ##    ##    ###          ## ####    ###    ########
  ## ##   ###   ## ###   ##   ## ##         ##  ##    ## ##       ##
 ##   ##  ####  ## ####  ##  ##   ##        ##  ##   ##   ##      ##
##     ## ## ## ## ## ## ## ##     ##       ##  ##  ##     ##     ##
######### ##  #### ##  #### ######### ##    ##  ##  #########     ##
##     ## ##   ### ##   ### ##     ## ##    ##  ##  ##      ##    ##
##     ## ##    ## ##    ## ##     ##  ######  #### ##      ##    ##
```

Feel free to choose your own method of storing data in the array but you have to print using loop(s). Printing manually by mentioning numeric value of index is NOT accepted.

Whatever output your program gives, if it is copied to MS word and viewed with “Courier New” font and a size of 10 points (pts), it should look similar to the above example.

Hint: Use “banner3” font with your name at http://www.network-science.de/ascii/