

## 107 - The Cat in the Hat

Time limit: 3.000 seconds

### Background

(An homage to Theodore Seuss Geisel)

The Cat in the Hat is a nasty creature,  
But the striped hat he is wearing has a rather nifty feature.

With one flick of his wrist he pops his top off.

Do you know what's inside that Cat's hat?  
A bunch of small cats, each with its own striped hat.

Each little cat does the same as line three,  
All except the littlest ones, who just say ``Why me?''

Because the littlest cats have to clean all the grime,  
And they're tired of doing it time after time!

### The Problem

A clever cat walks into a messy room which he needs to clean. Instead of doing the work alone, it decides to have its helper cats do the work. It keeps its (smaller) helper cats inside its hat. Each helper cat also has helper cats in its own hat, and so on. Eventually, the cats reach a smallest size. These smallest cats have no additional cats in their hats. These unfortunate smallest cats have to do the cleaning.

The number of cats inside each (non-smallest) cat's hat is a constant,  $N$ . The height of these cats-in-a-hat is  $\frac{1}{N+1}$  times the height of the cat whose hat they are in.

The smallest cats are of height one;  
these are the cats that get the work done.  
All heights are positive integers.

Given the height of the initial cat and the number of worker cats (of height one), find the number of cats that are not doing any work (cats of height greater than one) and also determine the sum of all the cats' heights (the height of a stack of all cats standing one on top of another).

### The Input

The input consists of a sequence of cat-in-hat specifications. Each specification is a single line consisting of two positive integers, separated by white space. The first integer is the height of the initial cat, and the second integer is the number of worker cats.

A pair of 0's on a line indicates the end of input.

## **The Output**

For each input line (cat-in-hat specification), print the number of cats that are not working, followed by a space, followed by the height of the stack of cats. There should be one output line for each input line other than the ``0 0" that terminates input.

## **Sample Input**

```
216 125
5764801 1679616
0 0
```

## **Sample Output**

```
31 671
335923 30275911
```

## 10042 - Smith Numbers

Time limit: 3.000 seconds

### Background

While skimming his phone directory in 1982, Albert Wilansky, a mathematician of Lehigh University, noticed that the telephone number of his brother-in-law H. Smith had the following peculiar property: The sum of the digits of that number was equal to the sum of the digits of the prime factors of that number. Got it? Smith's telephone number was 493-7775. This number can be written as the product of its prime factors in the following way:

$$4937775 = 3 \cdot 5 \cdot 5 \cdot 65837$$

The sum of all digits of the telephone number is  $4+9+3+7+7+7+5=42$ , and the sum of the digits of its prime factors is equally  $3+5+5+6+5+8+3+7=42$ . Wilansky was so amazed by his discovery that he named this type of numbers after his brother-in-law: Smith numbers.

As this observation is also true for every prime number, Wilansky decided later that a (simple and unsophisticated) prime number is not worth being a Smith number and he excluded them from the definition.

### Problem

Wilansky published an article about Smith numbers in the *Two Year College Mathematics Journal* and was able to present a whole collection of different Smith numbers: For example, 9985 is a Smith number and so is 6036. However, Wilansky was not able to give a Smith number which was larger than the telephone number of his brother-in-law. It is your task to find Smith numbers which are larger than 4937775.

### 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 smaller than  $10^9$ .

### Output

For every input value  $n$ , you are to compute the smallest Smith number which is larger than  $n$  and print each number on a single line. You can assume that such a number exists.

### Sample Input

```
1
4937774
```

### Sample Output

```
4937775
```

## 10044 - Erdos Numbers

Time limit: 3.000 seconds

### Background

The Hungarian Paul Erdős (1913-1996, speak as "Ar-dish") not only was one of the strangest mathematicians of the 20th century, he was also one of the most famous. He kept on publishing widely circulated papers up to a very high age and every mathematician having the honor of being a co-author to Erdős is well respected.

Not everybody got the chance to co-author a paper with Erdős, so many people were content if they managed to publish a paper with somebody who had published a scientific paper with Erdős. This gave rise to the so-called *Erdős numbers*. An author who has jointly published with Erdős had Erdős number 1. An author who had not published with Erdős but with somebody with Erdős number 1 obtained Erdős number 2, and so on.

### Problem

Today, nearly everybody wants to know which Erdős number he or she has. Your task is to write a program which computes Erdős numbers for a given set of scientists.

### Input

The first line of the input contains the number of scenarios.

The input for each scenario consists of a paper database and a list of names. It begins with the line

$P$   $N$

where  $P$  and  $N$  are natural numbers. Following this line are  $P$  lines containing descriptions of papers (this is the paper database). A paper appears on a line by itself and is specified in the following way:

```
Smith, M.N., Martin, G., Erdos, P.: Newtonian forms of prime factors
matrices
```

Note that umlauts like "ö" are simply written as "o". After the  $P$  papers follow  $N$  lines with names. Such a name line has the following format:

```
Martin, G.
```

### Output

For every scenario you are to print a line containing a string "Scenario  $i$ " (where  $i$  is the number of the scenario) and the author names together with their Erdős number of all authors in the list of names. The authors should appear in the same order as they appear in the list of names. The Erdős number is based on the papers in the paper database of this scenario. Authors which do not have any relation to Erdős via the papers in the database have Erdős number "infinity".

### Sample Input

```
1
4 3
Smith, M.N., Martin, G., Erdos, P.: Newtonian forms of prime factor
matrices
```

Erdos, P., Reisig, W.: Stuttering in petri nets  
Smith, M.N., Chen, X.: First order derivatives in structured programming  
Jablonski, T., Hsueh, Z.: Selfstabilizing data structures  
Smith, M.N.  
Hsueh, Z.  
Chen, X.

### **Sample Output**

Scenario 1  
Smith, M.N. 1  
Hsueh, Z. infinity  
Chen, X. 2

## 948 - Fibonaccimal Base

Time limit: 3.000 seconds

The well known Fibonacci sequence is obtained by starting with 0 and 1 and then adding the two last numbers to get the next one. For example the third number in the sequence is 1 ( $1=1+0$ ), the fourth is 2 ( $2=1+1$ ), the fifth is 3 ( $3=2+1$ ) and so on.

<b>i</b>	0	1	2	3	4	5	6	7	8	9
<b>Fib(i)</b>	0	1	1	2	3	5	8	13	21	34

**Figure 1** - The first numbers in the Fibonacci sequence

The sequence appears on many things in our life, in nature, and has a great significance. Among other things, do you know that all positive integer numbers can be represented as a sum of numbers in the Fibonacci sequence? More than that, all positive integers can be represented as a sum of a set of Fibonacci numbers, that is, numbers from the sequence, without repetition. For example:  $13$  can be the sum of the sets  $\{13\}$ ,  $\{5,8\}$  or  $\{2,3,8\}$  and  $17$  is represented by  $\{1,3,13\}$  or  $\{1,3,5,8\}$ . Since all numbers have this property (do you want to try to prove this for yourself?) this set could be a nice way to use as a "base" to represent the number. But, as we have seen, some numbers have more than one set whose sum is the number. How can we solve that? Simple! If we add the constraint that the sets cannot have two consecutive Fibonacci numbers, then we have a unique representation for each number! This restriction is because the sum of any two consecutive Fibonacci numbers is just the following Fibonacci number.

Now that we know all this we can prepare a nice way to represent any positive integer. We will use a binary sequence (just zeros and ones) to do that. For example,  $17 = 1 + 3 + 13$  (remember that no two consecutive Fibonacci numbers can be used). Let's write a zero for each Fibonacci number that is not used and one for each one that is used, starting at the right. Then,  $17 = 100101$ . See figure 2 for a detailed explanation. In this representation we should not have zeros at the left, this is, we should only write starting with the first one. In order for you to understand better, note that in this scheme, not using two consecutive Fibonacci numbers means that the binary sequence will not have two consecutive ones. When we use this representation for a number we say that we are using the Fibonaccimal base, and we write it like  $17 = 100101$  (*fib*).

<b>17 =</b>	1	0	0	1	0	1
<b>13+3+1 =</b>	<b>13</b>	<b>8</b>	<b>5</b>	<b>3</b>	<b>2</b>	<b>1</b>

**Figure 2** - Explaining the representation of 17 in Fibonaccimal base

### The Problem

Given a set of numbers in decimal base, your task is to write them in the Fibonaccimal base.

## ***Input***

The first line of input contains a single number  $N$ , representing the quantity of numbers that follow ( $1 \leq N \leq 500$ ).

Then follow exactly  $N$  lines, each one containing a single positive integer smaller than 100 000 000. These numbers can come in any order.

## ***Output***

You should output a single line for each of the  $N$  integers in the input, with the format "DEC\_BASE = FIB\_BASE (fib)". DEC\_BASE is the original number in decimal base and FIB\_BASE is its representation in Fibonaccimal base. See the sample output for an example.

### ***Example Input***

```
10
1
2
3
4
5
6
7
8
9
10
```

### ***Example Output***

```
1 = 1 (fib)
2 = 10 (fib)
3 = 100 (fib)
4 = 101 (fib)
5 = 1000 (fib)
6 = 1001 (fib)
7 = 1010 (fib)
8 = 10000 (fib)
9 = 10001 (fib)
10 = 10010 (fib)
```

**10419 - Sum-up the Primes**

Time limit: 4.000 seconds

We all know from Goldbach's conjecture that any even number greater than 2 can be expressed as a summation of two primes. Some odd numbers can also be expressed as summation of two primes. In this problem you will have to express a number as a summation of arbitrary number of primes less than 300. The conditions in detail are as follows:

- 1) You have to express a number  $N$  ( $N \leq 1000$ ) as a summation of  $t$  ( $t \leq 14$ ) primes.
- 2) Among the  $t$  primes any single odd primes can be present maximum two times. 2 can be present only once. For example,  $(5+5+3+3)$  is valid, but  $(3+3+3+7)$  or  $(2+2+3)$  is invalid according to this particular rule.
- 3) All the prime numbers used must be less than 300.
- 4) If there is more than one solution print the lexicographically smallest one.
- 5) If there is no such expression of primes print the string "No Solution."

**Input**

The input file contains less than 9340 lines of input. Each line contains two numbers  $N$  ( $0 < N \leq 1000$ ) and  $t$  ( $0 < t \leq 14$ ). The meaning of  $N$  and  $t$  are described in the problem statement. Input is terminated by a line where  $N=0$  and  $t=0$ . This line should not be processed.

**Output**

For each line of input produce a block of 2 lines. The first line of such a block contains the output serial as shown in the sample output. Next line contains the lexicographically smallest expression that sums up to  $N$ . There is no space between the operators and operands. If  $N$  cannot be expressed as a summation of  $t$  primes output "No Solution."

**Sample Input**

20 10

100 4

10 2

0 0

**Sample Output**

CASE 1:

No Solution.



CASE 2:

$11+11+17+61$

CASE 3:

$3+7$

**11346 - Probability**

Time limit: 1.000 seconds

**Time Limit: 1 sec****Memory Limit: 16MB**

Consider rectangular coordinate system and point  $L(X,Y)$  which is randomly chosen among all points in the area  $A$  which is defined in the following manner:  $A = \{(x,y) \mid x \text{ is from interval } [-a;a]; y \text{ is from interval } [-b;b]\}$ . What is the probability  $P$  that the area of a rectangle that is defined by points  $(0,0)$  and  $(X,Y)$  will be greater than  $S$ ?

**INPUT:**

The number of tests  $N \leq 200$  is given on the first line of input. Then  $N$  lines with one test case on each line follow. The test consists of 3 real numbers  $a > 0$ ,  $b > 0$  or  $S \geq 0$ .

**OUTPUT:**

For each test case you should output one number  $P$  and percentage "%" symbol following that number on a single line.  $P$  must be rounded to 6 digits after decimal point.

**SAMPLE INPUT:**

```
3
10 5 20
1 1 1
2 2 0
```

**SAMPLE OUTPUT:**

```
23.348371%
0.000000%
100.000000%
```

## 200 - Rare Order

Time limit: 3.000 seconds

A rare book collector recently discovered a book written in an unfamiliar language that used the same characters as the English language. The book contained a short index, but the ordering of the items in the index was different from what one would expect if the characters were ordered the same way as in the English alphabet. The collector tried to use the index to determine the ordering of characters (i.e., the collating sequence) of the strange alphabet, then gave up with frustration at the tedium of the task.

You are to write a program to complete the collector's work. In particular, your program will take a set of strings that has been sorted according to a particular collating sequence and determine what that sequence is.

### *Input*

The input consists of an ordered list of strings of uppercase letters, one string per line. Each string contains at most 20 characters. The end of the list is signalled by a line with the single character '#'. Not all letters are necessarily used, but the list will imply a complete ordering among those letters that are used.

### *Output*

Your output should be a single line containing uppercase letters in the order that specifies the collating sequence used to produce the input data file.

### *Sample Input*

```
XWY
ZX
ZXY
ZXW
YWWX
#
```

### *Sample Output*

```
XZYW
```

## 439 - Knight Moves

Time limit: 3.000 seconds

A friend of you is doing research on the *Traveling Knight Problem (TKP)* where you are to find the shortest closed tour of knight moves that visits each square of a given set of  $n$  squares on a chessboard exactly once. He thinks that the most difficult part of the problem is determining the smallest number of knight moves between two given squares and that, once you have accomplished this, finding the tour would be easy.

Of course you know that it is vice versa. So you offer him to write a program that solves the "difficult" part.

Your job is to write a program that takes two squares  $a$  and  $b$  as input and then determines the number of knight moves on a shortest route from  $a$  to  $b$ .

### Input Specification

The input file will contain one or more test cases. Each test case consists of one line containing two squares separated by one space. A square is a string consisting of a letter (a-h) representing the column and a digit (1-8) representing the row on the chessboard.

### Output Specification

For each test case, print one line saying "To get from  $xx$  to  $yy$  takes  $n$  knight moves."

### Sample Input

```
e2 e4
a1 b2
b2 c3
a1 h8
a1 h7
h8 a1
b1 c3
f6 f6
```

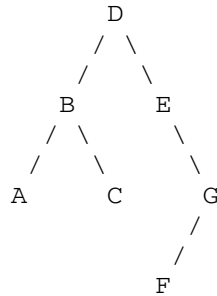
### Sample Output

```
To get from e2 to e4 takes 2 knight moves.
To get from a1 to b2 takes 4 knight moves.
To get from b2 to c3 takes 2 knight moves.
To get from a1 to h8 takes 6 knight moves.
To get from a1 to h7 takes 5 knight moves.
To get from h8 to a1 takes 6 knight moves.
To get from b1 to c3 takes 1 knight moves.
To get from f6 to f6 takes 0 knight moves.
```

## 536 - Tree Recovery

Time limit: 3.000 seconds

Little Valentine liked playing with binary trees very much. Her favorite game was constructing randomly looking binary trees with capital letters in the nodes. This is an example of one of her creations:



To record her trees for future generations, she wrote down two strings for each tree: a preorder traversal (root, left subtree, right subtree) and an inorder traversal (left subtree, root, right subtree).

For the tree drawn above the preorder traversal is DBACEGF and the inorder traversal is ABCDEFG.

She thought that such a pair of strings would give enough information to reconstruct the tree later (but she never tried it). Now, years later, looking again at the strings, she realized that reconstructing the trees was indeed possible, but only because she never had used the same letter twice in the same tree.

However, doing the reconstruction by hand, soon turned out to be tedious.

So now she asks you to write a program that does the job for her!

### **Input Specification**

The input file will contain one or more test cases. Each test case consists of one line containing two strings preord and inord, representing the preorder traversal and inorder traversal of a binary tree. Both strings consist of unique capital letters. (Thus they are not longer than 26 characters.) . Input is terminated by end of file.

### **Output Specification**

For each test case, recover Valentine's binary tree and print one line containing the tree's postorder traversal (left subtree, right subtree, root).

### **Sample Input**

```
DBACEGF ABCDEFG
BCAD CBAD
```

### **Sample Output**

```
ACBFGED
CDAB
```

## 10307 - Killing Aliens in Borg Maze

The Borg is an immensely powerful race of enhanced humanoids from the delta quadrant of the galaxy. The Borg collective is the term used to describe the group consciousness of the Borg civilization. Each Borg individual is linked to the collective by a sophisticated subspace network that insures each member is given constant supervision and guidance. Your task is to help the Borg (yes, really) by developing a program which helps the Borg to estimate the minimal cost of scanning a maze for the assimilation of aliens hiding in the maze, by moving in north, west, east, and south steps. The tricky thing is that the beginning of the search is conducted by a large group of over **100** individuals. Whenever an alien is assimilated, or at the beginning of the search, the group may split in two or more groups (but their consciousness is still collective.). The cost of searching a maze is defined as the total distance covered by all the groups involved in the search together. That is, if the original group walks five steps, then splits into two groups each walking three steps, the total distance is **11=5+3+3**.

### Input

On the first line of input there is one integer,  $N \leq 50$ , giving the number of test cases in the input. Each test case starts with a line containing two integers  $x, y$  such that  $1 \leq x, y \leq 50$ . After this,  $y$  lines follow, each which  $x$  characters. For each character, a space " " stands for an open space, a hash mark "#" stands for an obstructing wall, the capital letter "A" stand for an alien, and the capital letter "S" stands for the start of the search. The perimeter of the maze is always closed, i.e., there is no way to get out from the coordinate of the "S". At most **100** aliens are present in the maze, and everyone is reachable.

### Output

For every test case, output one line containing the minimal cost of a successful search of the maze leaving no aliens alive.

### Sample Input

```
2
6 5
#####
#A#A##
# # A#
#S  ##
#####
7 7
#####
#AAA###
#   A#
# S ###
#   #
#AAA###
#####
```

### Sample Output

```
8
11
```