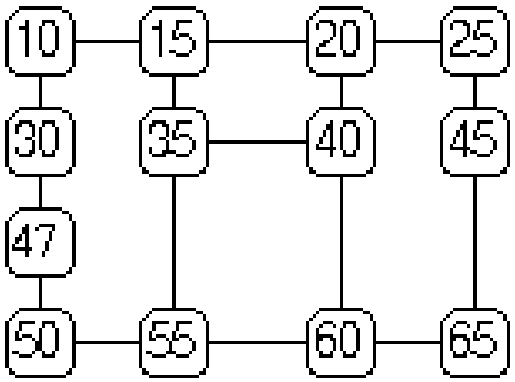
336 A Node Too Far

To avoid the potential problem of network messages (packets) looping around forever inside a network, each message includes a Time To Live (TTL) field. This field contains the number of nodes (stations, computers, etc.) that can retransmit the message, forwarding it along toward its destination, before the message is unceremoniously dropped. Each time a station receives a message it decrements the TTL field by 1. If the destination of the message is the current station, then the TTL field's value is ignored. However, if the message must be forwarded, and the decremented TTL field contains zero, then the message is not forwarded.

In this problem you are given the description of a number of networks, and for each network you are asked to determine the number of nodes that are not reachable given an initial node and TTL field value. Consider the following example network:



If a message with a TTL field of 2 was sent from node 35 it could reach nodes 15, 10, 55, 50, 40, 20 and 60. It could not reach nodes 30, 47, 25, 45 or 65, since the TTL field would have been set to zero on arrival of the message at nodes 10, 20, 50 and 60. If we increase the TTL field's initial value to 3, starting from node 35 a message could reach all except node 45.

Input and Output

There will be multiple network configurations provided in the input. Each network description starts with an integer NC specifying the number of connections between network nodes. An NC value of zero marks the end of the input data. Following NC there will be NC pairs of positive integers. These pairs identify the nodes that are connected by a communication line. There will be no more than one (direct) communication line between any pair of nodes, and no network will contain more than 30 nodes. Following each network configuration there will be multiple queries as to how many nodes are not reachable given an initial node and TTL field setting. These queries are given as a pair of integers, the first identifying the starting node and the second giving the initial TTL field setting. The queries are terminated by a pair of zeroes.

For each query display a single line showing the test case number (numbered sequentially from one), the number of nodes not reachable, the starting node number, and the initial TTL field setting. The sample input and output shown below illustrate the input and output format.

Sample Input

```
16
10 15   15 20   20 25   10 30   30 47   47 50   25 45   45 65
15 35   35 55   20 40   50 55   35 40   55 60   40 60   60 65
35  2   35  3    0  0

14
1 2   2 7   1 3   3 4   3 5   5 10   5 11
4 6   7 6   7 8   7 9   8 9   8  6   6 11
1 1   1 2   3 2   3 3   0 0

0
```

Sample Output

```
Case 1: 5 nodes not reachable from node 35 with TTL = 2.
Case 2: 1 nodes not reachable from node 35 with TTL = 3.
Case 3: 8 nodes not reachable from node 1 with TTL = 1.
Case 4: 5 nodes not reachable from node 1 with TTL = 2.
Case 5: 3 nodes not reachable from node 3 with TTL = 2.
Case 6: 1 nodes not reachable from node 3 with TTL = 3.
```

694 The Collatz Sequence

An algorithm given by Lothar Collatz produces sequences of integers, and is described as follows:

Step 1: Choose an arbitrary positive integer A as the first item in the sequence.

Step 2: If $A = 1$ then stop.

Step 3: If A is even, then replace A by $A/2$ and go to step 2.

Step 4: If A is odd, then replace A by $3 * A + 1$ and go to step 2.

It has been shown that this algorithm will always stop (in step 2) for initial values of A as large as 10^9, but some values of A encountered in the sequence may exceed the size of an integer on many computers. In this problem we want to determine the length of the sequence that includes all values produced until either the algorithm stops (in step 2), or a value larger than some specified limit would be produced (in step 4).

Input

The input for this problem consists of multiple test cases. For each case, the input contains a single line with two positive integers, the first giving the initial value of A (for step 1) and the second giving L, the limiting value for terms in the sequence. Neither of these, A or L, is larger than 2,147,483,647 (the largest value that can be stored in a 32-bit signed integer). The initial value of A is always less than L. A line that contains two negative integers follows the last case.

Output

For each input case display the case number (sequentially numbered starting with 1), a colon, the initial value for A, the limiting value L, and the number of terms computed.

Sample Input

```
3 100
34 100
75 250
27 2147483647
101 304
101 303
-1 -1
```

Sample Output

```
Case 1: A = 3, limit = 100, number of terms = 8
Case 2: A = 34, limit = 100, number of terms = 14
Case 3: A = 75, limit = 250, number of terms = 3
Case 4: A = 27, limit = 2147483647, number of terms = 112
Case 5: A = 101, limit = 304, number of terms = 26
Case 6: A = 101, limit = 303, number of terms = 1
```

UVa Online Judge

10260 Soundex

Soundex coding groups together words that appear to sound alike based on their spelling. For example, “can” and “khawn”, “con” and “gone” would be equivalent under Soundex coding.

Soundex coding involves translating each word into a series of digits in which each digit represents a letter:

1 represents B, F, P, or V
2 represents C, G, J, K, Q, S, X, or Z
3 represents D or T
4 represents L
5 represents M or N
6 represents R

The letters A, E, I, O, U, H, W, and Y are not represented in Soundex coding, and repeated letters with the same code digit are represented by a single instance of that digit. Words with the same Soundex coding are considered equivalent.

Input

Each line of input contains a single word, all upper case, less than 20 letters long.

Output

For each line of input, produce a line of output giving the Soundex code.

Sample Input

```
KHAWN
PFISTER
BOBBY
```

Sample Output

```
25
1236
11
```

UVa Online Judge

10324 Zeros and Ones

Given a string of 0's and 1's up to 1000000 characters long and indices i and j, you are to answer a question whether all characters between position $\min(i, j)$ and position $\max(i, j)$ (inclusive) are the same.

Input

There are multiple cases on input. The first line of each case gives a string of `0`'s and `1`'s. The next line contains a positive integer n giving the number of queries for this case. The next n lines contain queries, one per line. Each query is given by two non-negative integers, i and j. For each query, you are to print '`Yes`' if all characters in the string between position $\min(i, j)$ and position $\max(i, j)$ are the same, and '`No`' otherwise.

Output

Each case on output should start with a heading as in the sample below. The input ends with an empty string that is a line containing only the new line character, this string should not be processed. The input may also with end of file. So keep check for both.

Sample Input

```
0000011111
3
0 5
4 2
5 9
01010101010101010101010101111111111111111111111111111111111110000000000000000
5
4 4
25 60
1 3
62 76
24 62
1
1
0 0
```

Sample Output

```
Case 1:
No
Yes
Yes
Case 2:
Yes
Yes
No
Yes
```

```
No
Case 3:
Yes
```

417 Word Index

Encoding schemes are often used in situations requiring encryption or information storage/transmission economy. Here, we develop a simple encoding scheme that encodes particular types of words with five or fewer (lower case) letters as integers.

Consider the English alphabet {a,b,c,...,z}. Using this alphabet, a set of *valid* words are to be formed that are in a strict lexicographic order. In this set of *valid* words, the successive letters of a word are in a strictly ascending order; that is, later letters in a valid word are always *after* previous letters with respect to their positions in the alphabet list {a,b,c,...,z}. For example,

```
abc aep gwz
```

are all *valid* three-letter words, whereas

```
aab are cat
```

are not.

For each *valid* word associate an integer which gives the position of the word in the alphabetized list of words. That is:

```
a -> 1
b -> 2
.
.
z -> 26
ab -> 27
ac -> 28
.
.
az -> 51
bc -> 52
.
.
vwxyz -> 83681
```

Your program is to read a series of input lines. Each input line will have a single word on it, that will be from one to five letters long. For each word read, if the word is *invalid* give the number 0. If the word read is *valid*, give the word's position index in the above alphabetical list.

Input

The input consists of a series of single words, one per line. The words are at least one letter long and no more that five letters. Only the lower case alphabetic {a,b,...,z} characters will be used as input. The first letter of a word will appear as the first character on an input line.

The input will be terminated by end-of-file.

Output

The output is a single integer, greater than or equal to zero (0) and less than or equal 83681. The first digit of an output value should be the first character on a line. There is one line of output for each input line.

Sample Input

```
z
a
cat
vwxyz
```

Sample Output

```
26
1
0
83681
```

UVa Online Judge

10107 What is the Median?

Median plays an important role in the world of statistics. By definition, it is a value which divides an array into two equal parts. In this problem you are to determine the current median of some long integers. Suppose, we have five numbers {1,3,6,2,7}. In this case, 3 is the median as it has exactly two numbers on its each side. {1,2} and {6,7}. If there are even number of values like {1,3,6,2,7,8}, only one value cannot split this array into equal two parts, so we consider the average of the middle values {3,6}. Thus, the median will be (3+6)/2 = 4.5. In this problem, you have to print only the integer part, not the fractional. As a result, according to this problem, the median will be 4 !

Input

The input file consists of series of integers X ($0 \leq X < 2^{31}$) and total number of integers N is less than 10000. The numbers may have leading or trailing spaces.

Output

For each input print the current value of the median.

Sample Input

```
1
3
4
60
70
50
2
```

Sample Output

```
1
2
3
3
4
27
4
```

458 The Decoder

Write a complete program that will correctly decode a set of characters into a valid message. Your program should read a given file of a simple coded set of characters and print the exact message that the characters contain. The code key for this simple coding is a one for one character substitution based upon a *single arithmetic manipulation* of the printable portion of the ASCII character set.

Input and Output

For example: with the input file that contains:

```
1JKJ´pz´{ol´{yhklthyr´vm´{ol´Jvu{yvs´Kh{h´Jvywvyh{pvu5
1PIT´pz´h´{yhklthyr´vm´{ol´Pu{lyuh{pvuhs´I|zpulzz´Thjopul´Jvywvyh{pvu5
1KLJ´pz´{ol´{yhklthyr´vm´{ol´Kpnp{hs´Lx|pwtlu{´Jvywvyh{pvu5
```

your program should print the message:

```
*CDC is the trademark of the Control Data Corporation.
*IBM is a trademark of the International Business Machine Corporation.
*DEC is the trademark of the Digital Equipment Corporation.
```

Your program should accept all sets of characters that use the same encoding scheme and should print the actual message of each set of characters.

Sample Input

```
1JKJ´pz´{ol´{yhklthyr´vm´{ol´Jvu{yvs´Kh{h´Jvywvyh{pvu5
1PIT´pz´h´{yhklthyr´vm´{ol´Pu{lyuh{pvuhs´I|zpulzz´Thjopul´Jvywvyh{pvu5
1KLJ´pz´{ol´{yhklthyr´vm´{ol´Kpnp{hs´Lx|pwtlu{´Jvywvyh{pvu5
```

Sample Output

```
*CDC is the trademark of the Control Data Corporation.
*IBM is a trademark of the International Business Machine Corporation.
*DEC is the trademark of the Digital Equipment Corporation.
```

538 Balancing Bank Accounts

Once upon a time there was a large team coming home from the ACM World Finals. The fifteen travellers were confronted with a big problem:

In the previous weeks, there had been many money transactions between them: Sometimes somebody paid the entrance fees of a theme park for the others, somebody else paid the hotel room, another one the rental car, and so on.

So now the big calculation started. Some people had paid more than others, thus the individual bank accounts had to be balanced again. "Who has to pay whom how much?", that was the question.

As such a calculation is a lot of work, we need a program now that will solve this problem next year.

Input

The input file will contain one or more test cases.

Each test case starts with a line containing two integers: the number of travellers n ($2 \leq n \leq 20$) and the number of transactions t ($1 \leq t \leq 1000$). On the next n lines the names of the travellers are given, one per line. The names only consist of alphabetic characters and contain no whitespace. On the following t lines, the transactions are given in the format $name_1 name_2 amount$ where $name_1$ is the person who gave $amount$ dollars to $name_2$. The amount will always be a non-negative integer less than 10000.

Input will be terminated by two values of 0 for n and t.

Output

For each test case, first print a line saying "`Case #`i" where i is the number of the test case.

Then, on the following lines, print a list of transactions that reverses the transactions given in the input, i.e. balances the accounts again. Use the same format as in the input. Print a blank line after each test case, even after the last one.

Additional restrictions:

- Your solution must consist of **at most** $n - 1$ **transactions**.
- Amounts may not be negative, i.e. never output "`A B -20`", output "`B A 20`" instead.

If there is more than one solution satisfying these restrictions, anyone is fine.

Sample Input

```
2 1
Donald
Dagobert
Donald Dagobert 15
4 4
John
Mary
Cindy
Arnold
```

```
John Mary 100
John Cindy 200
Cindy Mary 40
Cindy Arnold 150
0 0
```

Sample Output

```
Case #1
Dagobert Donald 15

Case #2
Mary John 140
Cindy John 10
Arnold John 150
```

UVa Online Judge

10188 Automated Judge Script

The judges from the programming contests are known to be very mean and very lazy. We, judges, want less work and more Wrong Answers! So, we'd like you to help us and write an automated judge script to judge solution runs from teams all over the world. All you have to do is write a program which receives the standard solution and a team output and gives as answer one of the following messages: "*Accepted*", "*Presentation Error*" or "*Wrong Answer*". We define each one as:

Accepted: As we are very mean judges, we only want you to give '`Accepted`' as answer if the team output matches the standard solution integrally. That is, ALL characters must match and must be in the same order.

Presentation Error: We want you to give '`Presentation Error`' if all NUMERIC characters match (and in the same order) but there is at least one non-numeric character wrong (or in wrong order). For instance, '`15 0`' and '`150`' would receive a '`Presentation Error`', whereas '`15 0`' and '`1 0`' would not (it would receive '`Wrong Answer`', see bellow).

Wrong Answer: If the team output could not be classified as any of the two above, then you have no option but to give '`Wrong Answer`' as an answer!

Input

The input will consist of an arbitrary number of input sets. Each input set begins with a positive integer $n < 100$, alone in a line, which describes the number of lines of the standard solution. The next n lines contain the standard solution. Then there is a positive integer $m < 100$, alone in a line, which describes the number of lines of the team output. The next m lines contain the team output. The input is terminated by a value of $n = 0$, and should not be processed.

No line will have more than 120 characters.

Output

For each set you should output one of the following lines:

```
Run #x: Accepted
Run #x: Presentation Error
Run #x: Wrong Answer
```

Where x stands for the number of the input set (starting from 1).

Sample Input

```
2
The answer is: 10
The answer is: 5
2
The answer is: 10
The answer is: 5
2
The answer is: 10
The answer is: 5
2
```

```
The answer is: 10
The answer is: 15
2
The answer is: 10
The answer is:  5
2
The answer is: 10
The answer is: 5
3
Input Set #1: YES
Input Set #2: NO
Input Set #3: NO
3
Input Set #0: YES
Input Set #1: NO
Input Set #2: NO
1
1 0 1 0
1
1010
1
The judges are mean!
1
The judges are good!
0
```

Sample Output

```
Run #1: Accepted
Run #2: Wrong Answer
Run #3: Presentation Error
Run #4: Wrong Answer
Run #5: Presentation Error
Run #6: Presentation Error
```

UVa Online Judge

10487 Closest Sums

Given is a set of integers and then a sequence of queries. A query gives you a number and asks to find a sum of two distinct numbers from the set, which is closest to the query number.

Input

Input contains multiple cases.

Each case starts with an integer n ($1 < n \leq 1000$), which indicates, how many numbers are in the set of integer. Next n lines contain n numbers. Of course there is only one number in a single line. The next line contains a positive integer m giving the number of queries, $0 < m < 25$. The next m lines contain an integer of the query, one per line.

Input is terminated by a case whose $n = 0$. Surely, this case needs no processing.

Output

Output should be organized as in the sample below. For each query output one line giving the query value and the closest sum in the format as in the sample. Inputs will be such that no ties will occur.

Sample Input

```
5
3
12
17
33
34
3
1
51
30
3
1
2
3
3
1
2
3
3
1
2
3
3
4
5
6
0
```

Sample Output

```
Case 1:
Closest sum to 1 is 15.
Closest sum to 51 is 51.
Closest sum to 30 is 29.
Case 2:
Closest sum to 1 is 3.
Closest sum to 2 is 3.
Closest sum to 3 is 3.
Case 3:
Closest sum to 4 is 4.
Closest sum to 5 is 5.
Closest sum to 6 is 5.
```

497 Strategic Defense Initiative

"Commander! Commander! Please wake up commander!"

"... mmmph. What time is it?"

"4:07 am, Commander. The following message just arrived on the emergency zeta priority classified scrambler, marked your eyes only."

You grudgingly take the letter, rub the sleep from your eyes, fleetingly wish that the 'Backer closed at an earlier hour, and start to read.

```
``Dear StarWars SDI Commander,

    Bad news, buddy. Crazy Boris had a bit too much vodka last night
    and when he woke up this morning, instead of the snooze button
    on his alarm clock, he ... well, let me put it this way: we´ve got
    tons of nuclear missles flying this way. Unfortunately, all that
    we have is a chart of the altitudes at which the missles are
    flying, arranged by the order of arrivals. Go for it, buddy.
    Good luck.

                                            Secretary of Defense

    P.S. Hilly and Bill say hi.´´
```

To make things worse, you remeber that SDI has a fatal flaw due to the budget cuts. When SDI sends out missles to intercept the targets, every missle has to fly higher than the previous one. In other words, once you hit a target, the next target can only be among the ones that are flying at higher altitudes than the one you just hit.

For example, if the missles are flying toward you at heights of 1, 6, 2, 3, and 5 (arriving in that order), you can try to intercept the first two, but then you won't be able to get the ones flying at 2, 3, 5 because they are lower than 6. Your job is to hit as many targets as possible. So you have to quickly write a program to find the best sequence of targets that the flawed SDI program is going to destroy.

Russian war tactics are fairly strange; their generals are stickers for mathematical precision. Their missles will always be fired in a sequence such that there will only be *one* solution to the problem posed above.

Input and Output

Input to your program will consist of a sequence of integer altitudes, each on a separate line.

Output from your program should contain the total number of targets you can hit, followed by the altitudes of those targets, one per line, in the order of their arrivals.

Sample Input

```
1
6
2
3
5
```

Sample Output

```
Max hits: 4
1
2
3
5
```

584 Bowling

History

Bowling has been traced to articles found in the tomb of an Egyptian child buried in 5200 BC. The primitive implements included nine pieces of stone at which a stone "ball" was rolled, the ball having first to roll through an archway made of three pieces of marble.

Another ancient discovery was the Polynesian game of ula maika, which also used pins and balls of stone. The stones were to be rolled at targets 60 feet away, a distance which is still one of the basic regulations of tenpins.

Bowling at tenpins probably originated in Germany, not as a sport but as a religious ceremony. Martin Luther is credited with settling on nine as the ideal number of pins.

Tracing history reveals the game moved through Europe, the Scandinavian countries and finally to the United States, where the earliest known reference to bowling at pins in America was made by author Washington Irving about 1818 in *Rip Van Winkle*.

Although the game was being played throughout the world, rules were different almost everywhere, and even basic equipment was not the same. In fact, why and when the 10th pin was added from the European game of ninepins to the American game of tenpins is still a mystery.

Rules

A single bowling game consists of ten *frames*. The object in each frame is to roll a ball at ten bowling pins arranged in an equilateral triangle and to knock down as many pins as possible.

For each frame, a bowler is allowed a maximum of two *rolls* to knock down all ten pins. If the bowler knocks them all down on the first attempt, the frame is scored as a *strike*. If the bowler does not knock them down on the first attempt in the frame the bowler is allowed a second attempt to knock down the remaining pins. If the bowler succeeds in knocking the rest of the pins down in the second attempt, the frame is scored as a *spare*.

The score for a bowling game consists of sum of the scores for each frame. The score for each frame is the total number of pins knocked down in the frame, plus bonuses for strikes and spares. In particular, if a bowler scores a strike in a particular frame, the score for that frame is ten plus the sum of the next two rolls. If a bowler scores a spare in a particular frame, the score for that frame is ten plus the score of the next roll. If a bowler scores a strike in the tenth (final) frame, the bowler is allowed two more rolls. Similarly, a bowler scoring a spare in the tenth frame is allowed one more roll.

The maximum possible score in a game of bowling (strikes in all ten frames plus two extra strikes for the tenth frame strike) is 300.

Input

The input will consist of a sequence of bowling game scores. Each line will contain the scores for a single game, with the scores for each roll of the ball separated by a single space. The score for a single roll will be represented by a single character — either a number indicating the number of pins knocked down, a '`/`' for a spare or a '`X`' for a strike.

The end of input is indicated by a single line containing the text `Game Over` (terminated with a newline).

Output

Your program should output the total game score for each game in the input file. The game scores should be left justified and each score should be printed on a separate line. The order of the scores on the output should correspond to the order of the games on the input.

Sample Input

```
1 0 1 / 2 2 X 3 3 X 1 / 3 / X 1 2
1 0 1 / 2 2 X 3 3 X 1 / 3 / 1 / X 8 0
1 0 1 / 2 2 X 3 3 X 1 / 3 / 1 / 8 / 9
Game Over
```

Sample Output

```
108
121
120
```