CSE - 105 (spring 2013)

50 UVA PROBLEMS

Md.Shamsujjoha

XPLOSIVE

Nasif Ahmed

Email: nasif.002@gmail.com

Skype: nasif.02

LIST OF PROBLEMS

1. 272 - TEX Quotes

Time limit: 3.000 seconds

TeX Quotes

TeX is a typesetting language developed by Donald Knuth. It takes source text together with a few typesetting instructions and produces, one hopes, a beautiful document. Beautiful documents use `` and " to delimit quotations, rather than the mundane " which is what is provided by most keyboards. Keyboards typically do not have an oriented double-quote, but they do have a left-single-quote ` and a right-single-quote '. Check your keyboard now to locate the left-single-quote key ` (sometimes called the ``backquote key") and the right-single-quote key ' (sometimes called the ``apostrophe" or just ``quote"). Be careful not to confuse the left-single-quote ` with the ``backslash" key \. TeX lets the user type two left-single-quotes `` to create a left-double-quote `` and two right-single-quotes '' to create a right-double-quote ". Most typists, however, are accustomed to delimiting their quotations with the un-oriented double-quote ".

If the source contained

```
"To be or not to be," quoth the bard, "that is the question."
```

then the typeset document produced by TeX would not contain the desired form:

``To be or not to be," quoth the bard, ``that is the question."

In order to produce the desired form, the source file must contain the sequence:

```
``To be or not to be,'' quoth the bard, ``that is the question.''
```

You are to write a program which converts text containing double-quote (") characters into text that is identical except that double-quotes have been replaced by the two-character sequences required by TeX for delimiting quotations with oriented double-quotes. The double-quote (") characters should be replaced appropriately by either `` if the " opens a quotation and by '' if the " closes a quotation. Notice that the question of nested quotations does not arise: The first " must be replaced by ``, the next by '', the next by ``, the next by '', the next by ``, the next by '', and so on.

Input and Output

Input will consist of several lines of text containing an even number of double-quote (") characters. Input is ended with an end-of-file character. The text must be output exactly as it was input except that:

- the first " in each pair is replaced by two ` characters: `` and
- the second " in each pair is replaced by two ' characters: ''.

Sample Input

```
"To be or not to be," quoth the Bard, "that
is the question".
The programming contestant replied: "I must disagree.
To `C' or not to `C', that is The Question!"
```

Sample Output

```
``To be or not to be,'' quoth the Bard, ``that
is the question''.
The programming contestant replied: ``I must disagree.
To `C' or not to `C', that is The Question!''
```

2. 494 - Kindergarten Counting Game

Time limit: 3.000 seconds

Kindergarten Counting Game

Everybody sit down in a circle. Ok. Listen to me carefully.

``Woooooo, you scwewy wabbit!"

Now, could someone tell me how many words I just said?

Input and Output

Input to your program will consist of a series of lines, each line containing multiple words (at least one). A ``word" is defined as a consecutive sequence of letters (upper and/or lower case).

Your program should output a word count for each line of input. Each word count should be printed on a separate line.

Sample Input

```
Meep Meep!
I tot I taw a putty tat.
I did! I did! I did taw a putty tat.
Shsssssssssh ... I am hunting wabbits. Heh Heh Heh Heh ...
```

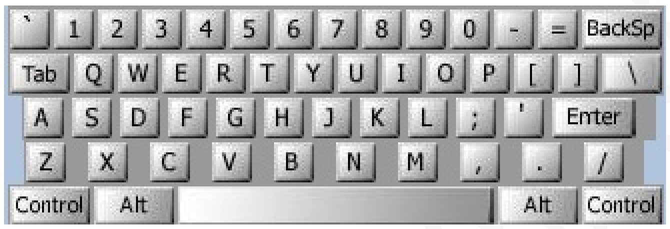
Sample Output

```
2
7
10
9
```

3. 10082 - WERTYU

Time limit: 3.000 seconds

Problem C: WERTYU



A common typing error is to place the hands on the keyboard one row to the right of the correct position. So "Q" is typed as "W" and "J" is typed as "K" and so on. You are to decode a message typed in this manner.

Input consists of several lines of text. Each line may contain digits, spaces, upper case letters (except Q, A, Z), or punctuation shown above [except back-quote (`)]. Keys labelled with words [*Tab, BackSp, Control*, etc.] are not represented in the input. You are to replace each letter or punction symbol by the one immediately to its left on the QWERTY keyboard shown above. Spaces in the input should be echoed in the output.

Sample Input

```
O S, GOMR YPFSU/
```

Output for Sample Input

```
I AM FINE TODAY.
```

4. 10018 - Reverse and Add

Time limit: 3.000 seconds

Reverse and Add

The Problem

The "reverse and add" method is simple: choose a number, reverse its digits and add it to the original. If the sum is not a palindrome (which means, it is not the same number from left to right and right to left), repeat this procedure.

For example:
195 Initial number
591

786
687

1473
3741

5214
4125

9339 Resulting palindrome

In this particular case the palindrome 9339 appeared after the 4th addition. This method leads to palindromes in a few step for almost all of the integers. But there are interesting exceptions. 196 is the first number for which no palindrome has been found. It is not proven though, that there is no such a palindrome.

Task:
You must write a program that gives the resulting palindrome and the number of iterations (additions) to compute the palindrome.

You might assume that all tests data on this problem:
- will have an answer,
- will be computable with less than 1000 iterations (additions),
- will yield a palindrome that is not greater than 4,294,967,295.

The Input

The first line will have a number N with the number of test cases, the next N lines will have a number P to compute its palindrome.

The Output

For each of the N tests you will have to write a line with the following data : minimum number of iterations (additions) to get to the palindrome and the resulting palindrome itself separated by one space.

Sample Input

```
3
195
265
750
```

Sample Output

```
4 9339
5 45254
3 6666
```

5. 10098 - Generating Fast

Time limit: 3.000 seconds

Problem C

Generating Fast, Sorted Permutation

Input: Standard Input
Output: Standard Output

Generating permutation has always been an important problem in computer science. In this problem you will have to generate the permutation of a given string in ascending order. Remember that your algorithm must be efficient.

Input

The first line of the input contains an integer n, which indicates how many strings to follow. The next n lines contain n strings. Strings will only contain alpha numerals and never contain any space. The maximum length of the string is 10.

Output

For each input string print all the permutations possible in ascending order. Not that the strings should be treated, as case sensitive strings and no permutation should be repeated. A blank line should follow each output set.

Sample Input

```
3
ab
abc
bca
```

Sample Output

```
ab
ba

abc
acb
bac
bca
cab
cba

abc
acb
bac
bca
cab
cba
```

Shahriar Manzoor

6. 458 - The Decoder

Time limit: 3.000 seconds

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.
```

7. 444 - Encoder and Decoder

Time limit: 3.000 seconds

Encoder and Decoder

Being in charge of the computer department of the Agency of International Espionage, you are asked to write a program that will allow a spy to encode and decode their messages.

You can assume a spy's message is at most 80 characters long, and it includes all the upper and lowercase letters of the alphabet plus the space, and any of the following characters:

```
!  ,  .  :  ;  ?
```

The following is an ASCII table of the valid characters in a message:

```
    "A"   65    "a"   97    " "   32
    "B"   66    "b"   98    "!"   33
     .           .          ","   44
     .           .          "."   46
     .           .          ":"   58
    "Y"   89    "y"   121   ";"   59
    "Z"   90    "z"   122   "?"   63
```

The algorithm that you should use to encode messages is to take the ASCII value of each character in the message, starting with the last character in the message and ending with the first character in the message. You should then add on to the coded message this ASCII value written in reverse order. For example, if the ASCII value is 123, the encoded message should contain the string "`321`". There should be no spaces separating the numbers in the encoded message.

Input and Output

The input file consists of one or more lines with a normal (not encoded) or encoded message each.

Output file must have the same number of lines with the corresponding encoded message or the decoded one, respectively.

Sample Input

```
abc
798999
Have a Nice Day !
```

Sample Output

```
998979
cba
332312179862310199501872379231018117927
```

8. 100 - The 3n + 1 problem

Time limit: 3.000 seconds

The 3*n* + 1 problem

Background

Problems in Computer Science are often classified as belonging to a certain class of problems (e.g., NP, Unsolvable, Recursive). In this problem you will be analyzing a property of an algorithm whose classification is not known for all possible inputs.

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←3n+1

5.       else n←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
```

9. 465 - Overflow

Time limit: 3.000 seconds

Overflow

Write a program that reads an expression consisting of two non-negative integer and an operator. Determine if either integer or the result of the expression is too large to be represented as a ``normal" signed integer (type `integer` if you are working Pascal, type `int` if you are working in C).

Input

An unspecified number of lines. Each line will contain an integer, one of the two operators `+` or `*`, and another integer.

Output

For each line of input, print the input followed by 0-3 lines containing as many of these three messages as are appropriate: ``first number too big'', ``second number too big'', ``result too big''.

Sample Input

```
300 + 3
9999999999999999999999 + 11
```

Sample Output

```
300 + 3
9999999999999999999999 + 11
first number too big
result too big
```

10. 10079 - Pizza Cutting

Time limit: 8.333 seconds

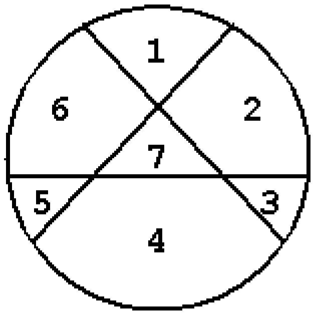
Problem E

Pizza Cutting

Input:standard input
Output: standard output

When someone calls Ivan lazy, he claims that it is his intelligence that helps him to be so. If his intelligence allows him to do something at less physical effort, why should he exert more? He also claims that he always uses his brain and tries to do some work at less effort; this is not his laziness, rather this is his intellectual smartness.

Once Ivan was asked to cut a pizza into seven pieces to distribute it among his friends. (Size of the pieces may not be the same. In fact, his piece will be larger than the others.) He thought a bit, and came to the conclusion that he can cut it into seven pieces by only three straight cuts through the pizza with a pizza knife. Accordingly, he cut the pizza in the following way (guess which one is Ivan's piece):



One of his friends, who never believed in Ivan's smartness, was startled at this intelligence. He thought, if Ivan can do it, why can't my computer? So he tried to do a similar (but not exactly as Ivan's, for Ivan will criticize him for stealing his idea) job with his computer. He wrote a program that took the number of straight cuts one makes through the pizza, and output a number representing the maximum number of pizza pieces it will produce.

Your job here is to write a similar program. It is ensured that Ivan's friend won't criticize you for doing the same job he did.

Input

The input file will contain a single integer N (0 <= N <= 210000000) in each line representing the number of straight line cuts one makes through the pizza. A negative number terminates the input.

Output

Output the maximum number of pizza pieces the given number of cuts can produce. Each line should contain only one output integer without any leading or trailing space.

Sample Input:

```
5
10
-100
```

Sample Output:

```
16
56
```

Rezaul Alam Chowdhury

11. 11150 - Cola

Time limit: 3.000 seconds

Problem C : Cola

Time limit: 10 seconds

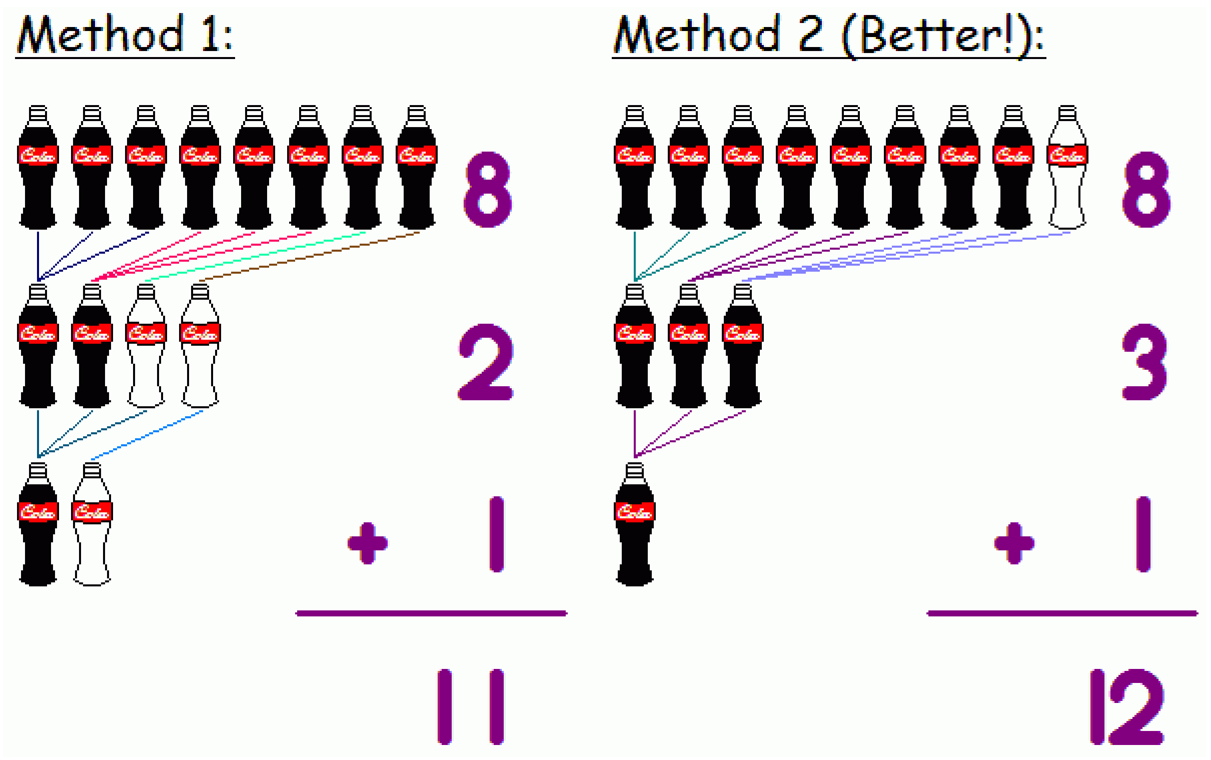
You see the following special offer by the convenience store:

" A bottle of Choco Cola for every 3 empty bottles returned "

Now you decide to buy some (say ***N***) bottles of cola from the store. You would like to know how you can get the most cola from them.

The figure below shows the case where ***N* = 8**. *Method 1* is the standard way: after finishing your 8 bottles of cola, you have 8 empty bottles. Take 6 of them and you get 2 new bottles of cola. Now after drinking them you have 4 empty bottles, so you take 3 of them to get yet another new cola. Finally, you have only 2 bottles in hand, so you cannot get new cola any more. Hence, you have enjoyed 8 + 2 + 1 = 11 bottles of cola.

You can actually do better! In *Method 2*, you first borrow an empty bottle from your friend (?! Or the storekeeper??), then you can enjoy 8 + 3 + 1 = 12 bottles of cola! Of course, you will have to return your remaining empty bottle back to your friend.



Input

Input consists of several lines, each containing an integer ***N*** ($1 \leq N \leq 200$).

Output

For each case, your program should output the maximum number of bottles of cola you can enjoy. You may borrow empty bottles from others, but if you do that, make sure that you have enough bottles afterwards to return to them.

Sample Input
```
8
```

Sample Output
```
12
```

Note: Drinking too much cola is bad for your health, so... don't try this at home!! :-)

Idea from a traditional IQ challenge question.
Special Thanks: Jonathan Mak

12. 10055 - Hashmat the Brave Warrior
Time limit: 3.000 seconds

Problem A
Hashmat the brave warrior
Input: standard input
Output: standard output

Hashmat is a brave warrior who with his group of young soldiers moves from one place to another to fight against his opponents. Before fighting he just calculates one thing, the difference between his soldier number and the opponent's soldier number. From this difference he decides whether to fight or not. Hashmat's soldier number is never greater than his opponent.

Input
The input contains two integer numbers in every line. These two numbers in each line denotes the number of soldiers in Hashmat's army and his opponent's army or vice versa. The input numbers are not greater than 2^32. Input is terminated by End of File.

Output
For each line of input, print the difference of number of soldiers between Hashmat's army and his opponent's army. Each output should be in seperate line.

Sample Input:
```
10 12
10 14
100 200
```

Sample Output:

```
2
4
100
```

Shahriar *Manzoor*
16-12-2000

13. 10071 - Back to High School Physics

Time limit: 3.000 seconds

Problem B

Back to High School Physics

Input: standard input
Output: standard output

A particle has initial velocity and constant acceleration. If its velocity after certain time is v then what will its displacement be in twice of that time?

Input

The input will contain two integers in each line. Each line makes one set of input. These two integers denote the value of v (-100 <= v <= 100) and t(0<=t<= 200) (t means at the time the particle gains that velocity)

Output

For each line of input print a single integer in one line denoting the displacement in double of that time.

Sample Input

```
0 0
5 12
```

Sample Output

```
0
120
```

Shahriar Manzoor

14. 10469 - To Carry or not to Carry

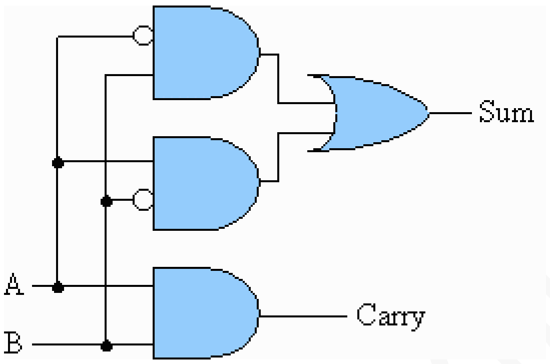
Time limit: 3.000 seconds

Return of the Aztecs

Problem G: To Carry or not to Carry

Time Limit: 1 second

Memory Limit: 32 MB



6+9=15 seems okay. But how come 4+6=2?

You see, Mofiz had worked hard throughout his digital logic course, but when he was asked to implement a 32 bit adder for the laboratory exam, he did some mistake in the design part. After tracing the design for half an hour, he found his flaw!! He was doing bitwise addition but his carry bit always had zero output. Thus,

```
 4 = 00000000 00000000 00000000 00000100
+6 = 00000000 00000000 00000000 00000110
----------------------------------------
 2 = 00000000 00000000 00000000 00000010
```

Its a good thing that he finally found his mistake, but it was too late. Considering his effort throughout the course, the instructor gave him one more chance. Mofiz has to write an efficient program that would take **2 unsigned 32 bit decimal numbers** as input, and produce an **unsigned 32 bit decimal number** as the output adding in the same was as his circuit does.

Input

In each line of input there will be a pair of integer separated by a single space. Input ends at EOF.

Output

For each line of input, output one line -- the value after adding the two numbers in the "Mofiz way".

Sample Input

```
4 6
6 9
```

Sample Output

```
2
15
```

Problem setter: Monirul Hasan (Tomal), CSE Dept, Southeast University, Bangladesh

"that you used cut and paste there is really nerdy." -- *goose*
"really? i use cut and paste all the time." -- *trip*

Collected from: nerd quotes

15. 10773 - Back to Intermediate Math

Time limit: 3.000 seconds

Back to Intermediate Math

Input: standard input
Output: standard output
Time Limit: 1 second

Umm! So, you claim yourself as an intelligent one? Let me check. As, computer science students always insist on optimization whenever possible, I give you an elementary problem of math to optimize.

You are trying to cross a river of width ***d*** **meters**. You are given that, the river flows at ***v*** **ms**$^{-1}$ and you know that you can speed up the boat in ***u*** **ms**$^{-1}$. There may be two goals how to cross the river: One goal (called fastest path) is to cross it in fastest time, and it does not matter how far the flow of the river takes the boat. The other goal (called shortest path) is to steer the boat in a direction so that the flow of the river doesn't take the boat away, and the boat passes the river in a line perpendicular to the boarder of the river. Is it always possible to have two different paths, one to pass at shortest time and the other at shortest path? If possible then, what is the difference (Let ***P*** **s**) between the times needed to cross the river in the different ways?

Input

The first line in the input file is an integer representing the number of test cases. Each of the test cases follows below. Each case consists three real numbers (all are nonnegative, ***d*** is positive) denoting the value of ***d***, ***v*** and ***u*** respectively.

Output

For each test case, first print the serial number of the case, a colon, an space and then print "***can't determine***" (without the quotes) if it is not possible to find different paths as stated above, else print the value of ***P*** corrected to three digits after decimal point. Check the sample input & output.

Sample Input

```
3
8 5 6
1 2 3
1 5 6
```

Sample Output

```
Case 1: 1.079
Case 2: 0.114
Case 3: 0.135
```

Problem setter: Anupam Bhattacharjee, CSE, BUET
Thanks to Adrian for alternate solution and clarification

**"~~ Learn from yesterday, live for today, hope for tomorrow.
The important thing is to not stop questioning. ~~"**

16. 11805 - Bafana Bafana

Time limit: 1.000 seconds

B	Bafana Bafana

Team practice is very important not only for programming contest but also for football. By team practice players can learn cooperating with team mates. For playing as a team improvement of passing skill is very important. Passing is a great way of getting the ball upfield and reduces the risk of giving the ball away.

Carlos Alberto Parreira, the coach of Bafana Bafana, also wants his players to practice passing a lot. That's why, while in the training camp for soccer world cup 2010, every day he asks all of the players who are present in practice to stand in a circle and practice passing. If **N** players are in practice, he gives each of the players a distinct number from 1 to **N**, and asks them to stand sequentially, so that player 2 will stand in right side of player 1 and player 3 will stand in right side of player 2, and so on. As they are in a circle, player 1 will stand right to player N.

The rule of passing practice is, Parreira will give the ball to player **K**, and practice will start. Practice will come to an end after **P** passes. In each pass, a player will give the ball to his partner who is in his immediate right side. After **P** passes, the player who owns the ball at that moment will give the ball back to Parreira.

Parreira wants to be ensured that his players practice according the rule. So he wants a program which will tell him which player will give him the ball back. So after taking the ball from the same person he can be happy that, the players play according to the rules. Otherwise he will ask them to start from beginning.

Input

Input starts with an integer **T** (**T** <= 1000), the number of test cases. Each test case will contain three integers, **N** (2 <= **N** <= 23), **K** (1 <= **K** <= **N**), **P**(1<=**P**<=200).

Output

For each test case, output a single line giving the case number followed by the Bafana player number who will give the ball to Parreira. See sample output for exact format.

Sample Input	Sample Output

`3`
`5 2 5`
`6 3 5`
`4 1 3`
	`Case 1: 2`
	`Case 2: 2`
	`Case 3: 4`

Problemsetter: Md. Arifuzzaman Arif, Special Thanks: Muntasir Khan, Sohel Hafiz

17. 299 - Train Swapping

Time limit: 3.000 seconds

Train Swapping

At an old railway station, you may still encounter one of the last remaining ``train swappers". A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains.

Once the carriages are arranged in the optimal order, all the train driver has to do, is drop the carriages off, one by one, at the stations for which the load is meant.

The title ``train swapper" stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right.

The first train swapper had discovered that the bridge could be operated with at most two carriages on it. By rotating the bridge 180 degrees, the carriages switched place, allowing him to rearrange the carriages (as a side effect, the carriages then faced the opposite direction, but train carriages can move either way, so who cares).

Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed, is a routine which decides for a given train the least number of swaps of two adjacent carriages necessary to order the train. Your assignment is to create that routine.

Input Specification

The input contains on the first line the number of test cases (*N*). Each test case consists of two input lines. The first line of a test case contains an integer *L*, determining the length of the train (0<=L<=50). The second line of a test case contains a permutation of the numbers 1 through *L*, indicating the current order of the carriages. The carriages should be ordered such that carriage 1 comes first, then 2, etc. with carriage *L* coming last.

Output Specification

For each test case output the sentence: '`Optimal train swapping takes Sswaps.`' where *S* is an integer.

Example Input

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

Example Output

```
Optimal train swapping takes 1 swaps.
Optimal train swapping takes 6 swaps.
Optimal train swapping takes 1 swaps.
```

18. 11588 - Image Coding

Time limit: 1.000 seconds

B

Image Coding

Input: Standard Input

Output: Standard Output

Communicating images over the internet is a costly process, thanks to the limits in available bandwidth. This problem is resolved to an extent by employing point of interest image coding. To be more specific, the point of interest image coding assumes that, some part of the image is most important, encodes this region with high quality & the other regions of the image with low quality, thus reducing file size.

You are to implement a very simple case of point of interest image coding. Assume an image file is represented by a grid of letters from 'A' to 'Z', each letter representing one pixel. All pixels represented by the same letter represents a same region in the image whereas two different letters stand for two different regions. The region with maximum number of letters is to be considered as the most important region. If there are more than one region tied for the maximum, treat all of them as most important. Every single letter in the most important region(s) is encoded as **M** byte data while each non-important region letter is encoded as **N** bytes. You are to determine the image file size. You can assume the image file consists of image data only i.e. no image header, title, meta information etc.

Input

The input file starts with **X,** the number of test cases (**1 ≤ X ≤ 50). X** cases follows. Each test case begins with 4 integers in a line, **R (1 ≤ R ≤ 20), C(1 ≤ C ≤ 20), M(1 ≤M ≤ 10) & N(1 ≤ N ≤ 10)** where **R** & **C** are the number of rows & columns in the pixel grid respectively. **M** is the number of bytes for each important pixel & **N** is the number of bytes for each non-important pixel. The following **R** lines describe the pixel grid. Each of these lines contain **C** letters between **'A'** & **'Z'** (inclusive)

Output

For each test case, print a line in the format **"Case x: y"** where **x** is the case number & **y** is the number of bytes in the image file.

Sample Input	Output for Sample Input
1 5 4 2 1 ABCD ABCA EFAC BCAG AZIP	Case 1: 26

Explanation of sample I / O: The image contains 10 regions namely 'A', 'B', 'C', 'D', 'E', 'F', 'G', Z', 'I' & 'P'. Region 'A' has the highest frequency of 6. So, this is the most important region. We encode 6 letters of this region with 2 bytes each & each of the remaining 14 letters with 1 byte. That makes the total file size = 6 * 2 + 14 * 1 = 26 bytes.

Problemsetter: Mohammad Mahmudur Rahman

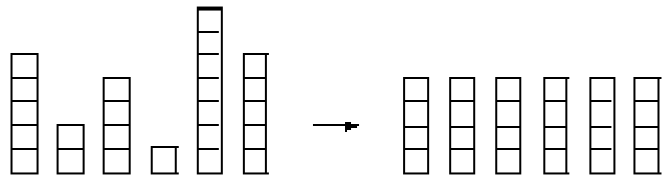
Special Thanks to: Md. Arifuzzaman Arif

19. 591 - Box of Bricks

Time limit: 3.000 seconds

Box of Bricks

Little Bob likes playing with his box of bricks. He puts the bricks one upon another and builds stacks of different height. ``Look, I've built a wall!", he tells his older sister Alice. ``Nah, you should make all stacks the same height. Then you would have a real wall.", she retorts. After a little con- sideration, Bob sees that she is right. So he sets out to rearrange the bricks, one by one, such that all stacks are the same height afterwards. But since Bob is lazy he wants to do this with the minimum number of bricks moved. Can you help?



Input

The input consists of several data sets. Each set begins with a line containing the number *n* of stacks Bob has built. The next line contains *n* numbers, the heights h_i of the *n* stacks. You may assume 1<=n<=50 and $1 \leq h_i \leq 100$.

The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height.

The input is terminated by a set starting with $n = 0$. This set should not be processed.

Output

For each set, first print the number of the set, as shown in the sample output. Then print the line ``The minimum number of moves is *k*.", where *k* is the minimum number of bricks that have to be moved in order to make all the stacks the same height.

Output a blank line after each set.

Sample Input

```
6
5 2 4 1 7 5
0
```

Sample Output

```
Set #1
The minimum number of moves is 5.
```

Miguel A. Revilla
1998-03-10

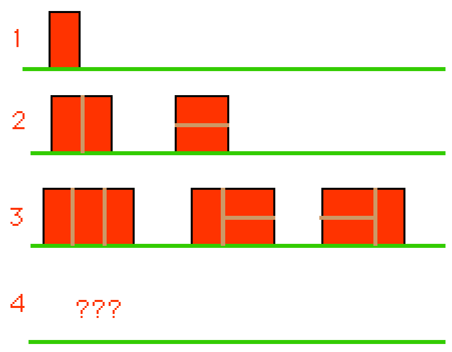
20. 900 - Brick Wall Patterns

Time limit: 3.000 seconds

Problem A

Brick Wall Patterns

If we want to build a brick wall out of the usual size of brick which has a length twice as long as its height, and if our wall is to be two units tall, we can make our wall in a number of patterns, depending on how long we want it. From the figure one observe that:



- There is just one wall pattern which is 1 unit wide - made by putting the brick on its end.
- There are 2 patterns for a wall of length 2: two side-ways bricks laid on top of each other and two bricks long-ways up put next to each other.
- There are three patterns for walls of length 3.

How many patterns can you find for a wall of length 4? And, for a wall of length 5?

Problem

Your task is to write a program that given the length of a wall, determines how many patterns there may be for a wall of that length.

Intput

Your program receives a sequence of positive integers, one per line, each representing the length of a wall. The maximum value for the wall is length 50. The input terminates with a 0.

Output

For each wall length given in the input, your program must output the corresponding number of different patterns for such a wall in a separate line.

Sample Intput

```
1
2
3
0
```

Sample Output

```
1
2
3
```

Fernando Silva, CPUP'2003 Round 1
Concurso de Programa��o da Universidade do Porto

21. 386 - Perfect Cubes

Time limit: 3.000 seconds

Perfect Cubes

For hundreds of years Fermat's Last Theorem, which stated simply that for $n > 2$ there exist no integers $a, b, c > 1$ such that $a^n = b^n + c^n$, has remained elusively unproven. (A recent proof is believed to be correct, though it is still undergoing scrutiny.) It is possible, however, to find integers greater than 1 that satisfy the ``perfect cube'' equation $a^3 = b^3 + c^3 + d^3$ (e.g. a quick calculation will show that the equation $12^3 = 6^3 + 8^3 + 10^3$ is indeed true). This problem requires that you write a program to find all sets of numbers $\{a, b, c, d\}$ which satisfy this equation for $a \le 200$.

Output

The output should be listed as shown below, one perfect cube per line, in non-decreasing order of *a* (i.e. the lines should be sorted by their a values). The values of *b*,*c*, and *d* should also be listed in non-decreasing order on the line itself. There do exist several values of a which can be produced from multiple distinct sets of *b*, *c*, and *d*triples. In these cases, the triples with the smaller *b* values should be listed first.

The first part of the output is shown here:

```
Cube = 6, Triple = (3,4,5)
Cube = 12, Triple = (6,8,10)
Cube = 18, Triple = (2,12,16)
Cube = 18, Triple = (9,12,15)
Cube = 19, Triple = (3,10,18)
Cube = 20, Triple = (7,14,17)
Cube = 24, Triple = (12,16,20)
```

Note: The programmer will need to be concerned with an efficient implementation. The official time limit for this problem is 2 minutes, and it is indeed possible to write a solution to this problem which executes in under 2 minutes on a 33 MHz 80386 machine. Due to the distributed nature of the contest in this region, judges have been instructed to make the official time limit at their site the greater of 2 minutes or twice the time taken by the judge's solution on the machine being used to judge this problem.

22. 10341 - Solve It

Time limit: 3.000 seconds

Problem F

Solve It

Input: standard input

Output: standard output

Time Limit: 1 second

Memory Limit: 32 MB

Solve the equation:

$$p*e^{-x} + q*\sin(x) + r*\cos(x) + s*\tan(x) + t*x^2 + u = 0$$

where $0 <= x <= 1$.

Input

Input consists of multiple test cases and terminated by an EOF. Each test case consists of 6 integers in a single line: p, q, r, s, t and u (where $0 <= p,r <= 20$ and $-20 <= q,s,t <= 0$). There will be maximum 2100 lines in the input file.

Output

For each set of input, there should be a line containing the value of x, correct upto 4 decimal places, or the string "`No solution`", whichever is applicable.

Sample Input

```
0 0 0 0 -2 1
1 0 0 0 -1 2
1 -1 1 -1 -1 1
```

Sample Output

```
0.7071
No solution
0.7554
```

Mustaq Ahmed

23. 10035 - Primary Arithmetic

Time limit: 3.000 seconds

Problem B: Primary Arithmetic

Children are taught to add multi-digit numbers from right-to-left one digit at a time. Many find the "carry" operation - in which a 1 is carried from one digit position to be added to the next - to be a significant challenge. Your job is to count the number of carry operations for each of a set of addition problems so that educators may assess their difficulty.

Input

Each line of input contains two unsigned integers less than 10 digits. The last line of input contains 0 0.

Output

For each line of input except the last you should compute and print the number of carry operations that would result from adding the two numbers, in the format shown below.

Sample Input

```
123 456
555 555
123 594
0 0
```

Sample Output

```
No carry operation.
3 carry operations.
1 carry operation.
```

24. 11495 - Bubbles and Buckets

Time limit: 3.000 seconds

Bubbles and Buckets

The Problem

Andrea, Carlos and Marcelo are close friends and spend their weekends by the swimming pool. While Andrea gets a suntan, both friends play *Bubbles*. Andrea, a very smart computer scientist, has already told them that she does not understand why they spend so much time playing a game so simple.

Using her laptop, Carlos and Marcelo generate a random integer *N* and a sequence, also random, which is a permutation from *1, 2, ..., N*.

The game then begins. The players play by turns, and at each turn a player makes a move. Marcelo is always the first to play.

A move consists of choosing one pair of consecutive elements that are out of order in the sequence, and swapping both elements. For example, given the sequence *1, 5, 3, 4, 2,* a player may swap *3* and *5* or *4* and *2,* but cannot swap *3* and *4* nor *5* and *2.* Continuing with the example, if the player decides to swap *5* and *3,* the new sequence will be *1, 3, 5, 4, 2.*

Sooner or later, the sequence will be sorted. The player that cannot make a move loses. Andrea, with disdain, always says that it would be simpler to play Odd or Even, to the same effect. Your mission, in case you decide to accept it, is to determine who wins the game, given the initial permutation *P*.

The Input

The input contains several test cases. Each test case is composed of a single line, in which all integers are separated by one space. Each line contains an integer *N* ($2 \le N \le$

10^5), followed by the initial sequence $P = (X_1, X_2, \ldots, X_N)$ of N distinct integers, with $1 \leq X_i \leq \text{N}$ for $1 \leq i \leq N$.

The end of input is indicated by a line containing only one zero.

The Output

For each test case in the input, your program must print a single line, containing the name of the winner, equal to `Carlos` or `Marcelo`.

Sample Input

```
5 1 5 3 4 2
5 5 1 3 4 2
5 1 2 3 4 5
6 3 5 2 1 4 6
5 5 4 3 2 1
6 6 5 4 3 2 1
0
```

Sample Output

```
Marcelo
Carlos
Carlos
Carlos
Carlos
Marcelo
```

25. 136 - Ugly Numbers

Time limit: 3.000 seconds

Ugly Numbers

Ugly numbers are numbers whose only prime factors are 2, 3 or 5. The sequence

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...

shows the first 11 ugly numbers. By convention, 1 is included.

Write a program to find and print the 1500'th ugly number.

Input and Output

There is no input to this program. Output should consist of a single line as shown below, with <number> replaced by the number computed.

Sample output

```
The 1500'th ugly number is <number>.
```

26. 11388 - GCD LCM

Time limit: 1.000 seconds

IIUC ONLINE CONTEST 2008

Problem D: GCD LCM

Input: standard input
Output: standard output

The GCD of two positive integers is the largest integer that divides both the integers without any remainder. The LCM of two positive integers is the smallest positive integer that is divisible by both the integers. A positive integer can be the GCD of many pairs of numbers. Similarly, it can be the LCM of many pairs of numbers. In this problem, you will be given two positive integers. You have to output a pair of numbers whose GCD is the first number and LCM is the second number.

Input

The first line of input will consist of a positive integer **T**. **T** denotes the number of cases. Each of the next **T** lines will contain two positive integer, **G** and **L.**

Output

For each case of input, there will be one line of output. It will contain two positive integers **a** and **b**, **a ≤ b**, which has a GCD of **G** and LCM of **L.** In case there is more than one pair satisfying the condition, output the pair for which **a** is minimized. In case there is no such pair, output -1.

Constraints

- **T ≤ 100**
- Both **G** and **L** will be less than $\mathbf{2^{31}}$.

Sample Input	Output for Sample Input

`2`
 	
`1 2`
 	
`3 4`
	`1 2`
	
	`-1`

Problem setter: Shamim Hafiz

27. 369 - Combinations

Time limit: 3.000 seconds

Combinations

Computing the exact number of ways that N things can be taken M at a time can be a great challenge when N and/or M become very large. Challenges are the stuff of contests. Therefore, you are to make just such a computation given the following:

GIVEN:

$$5 \leq N \leq 100, \qquad \text{and} \qquad 5 \leq M \leq 100, \qquad \text{and} \qquad M \leq N$$

Compute the **EXACT** value of:

$$C = \frac{N!}{(N-M)! \times M!}$$

You may assume that the final value of C will fit in a 32-bit Pascal LongInt or a C long.

For the record, the exact value of 100! is:

```
     93,326,215,443,944,152,681,699,238,856,266,700,490,715,968,264,381,621,
        468,592,963,895,217,599,993,229,915,608,941,463,976,156,518,286,253,

697,920,827,223,758,251,185,210,916,864,000,000,000,000,000,000,000,000
```

Input and Output

The input to this program will be one or more lines each containing zero or more leading spaces, a value for *N*, one or more spaces, and a value for *M*. The last line of the input file will contain a dummy *N*, *M* pair with both values equal to zero. Your program should terminate when this line is read.

The output from this program should be in the form:

N `things taken` *M* `at a time is` *C* `exactly.`

Sample Input

```
     100  6
      20  5
      18  6
       0  0
```

Sample Output

```
100 things taken 6 at a time is 1192052400 exactly.
20 things taken 5 at a time is 15504 exactly.
18 things taken 6 at a time is 18564 exactly.
```

28. 10038 - Jolly Jumpers

Time limit: 3.000 seconds

Problem E: Jolly Jumpers

A sequence of *n > 0* integers is called a *jolly jumper* if the absolute values of the difference between successive elements take on all the values 1 through *n-1*. For instance,

```
1 4 2 3
```

is a jolly jumper, because the absolutes differences are 3, 2, and 1 respectively. The definition implies that any sequence of a single integer is a jolly jumper. You are to write a program to determine whether or not each of a number of sequences is a jolly jumper.

Input

Each line of input contains an integer *n* <= 3000 followed by *n* integers representing the sequence.

Output

For each line of input, generate a line of output saying "Jolly" or "Not jolly".

Sample Input

```
4 1 4 2 3
5 1 4 2 -1 6
```

Sample Output

```
Jolly
Not jolly
```

29. 11799 - Horror Dash

Time limit: 1.000 seconds

F	Horror Dash	
	Input	Standard Input
	Output	Standard Output

It is that time of the year again! Colorful balloons and brightly colored banners spread out over your entire neighborhood for just this one occasion. It is the annual clown's festival at your local school. For the first time in their lives, students from the school try their hands at being the best clown ever. Some walk on long poles, others try to keep a crowd laughing for the day with stage comedy, while others still try out their first juggling act - some 'master clowns' even teach these juggling tricks to visitors at the festival.

As part of the festival, there is a unique event known as the "Horror Dash". At this event, ***N*** **(1≤*N*≤100)** students dressed in the scariest costumes possible start out in a race to catch a poor clown running on the same track. The clown trips over, loses his mind, and does all sorts of comical acts all while being chased round and round on the track. To keep the event running for as long as possible, the clown must run fast enough not to be caught by any of the scary creatures. However, to keep the audience on the edge of their seats, the clown must not run too fast either. This is where you are to help. Given the speed of every scary creature, you are to find out the minimum speed that the clown must maintain so as not to get caught even if they keep on running forever.

Input

The first line of input contains a single integer ***T*** **(*T*≤50)**, the number of test cases. This line is followed by ***T*** input cases. Each input case is on a single line of space-separated integers. The first of these integers is ***N***, the number of students acting as scary creatures. The rest of the line has ***N*** more integers, c_0, c_1, ..., c_n, each representing the speed of a creature in meters per second **(1≤c_i≤10000 for each*i*)**. You can assume that they are always running in the same direction on the track.

Output

There should be a single line of output for each test case, formatted as "`Case c: s`". Here, ***c*** represents the serial number of the input case, starting with 1, while ***s***represents the required speed of the clown, in meters per second.

Sample Input	**Sample Output**

2
5 9 3 5 2 6
1 2
	Case 1: 9
	Case 2: 2

30. 484 - The Department of Redundancy Department

Time limit: 3.000 seconds

The Department of Redundancy Department

Write a program that will remove all duplicates from a sequence of integers and print the list of unique integers occuring in the input sequence, along with the number of occurences of each.

Input

The input file will contain a sequence of integers (positive, negative, and/or zero). The input file may be arbitrarily long.

Output

The output for this program will be a sequence of ordered pairs, separated by newlines. The first element of the pair must be an integer from the input file. The second element must be the number of times that that particular integer appeared in the input file. The elements in each pair are to be separated by space characters. The integers are to appear in the order in which they were contained in the input file.

Sample Input

```
3 1 2 2 1 3 5 3 3 2
```

Sample Output

```
3 4
1 2
2 3
5 1
```

31. 628 - Passwords

Time limit: 3.000 seconds

Passwords

Having several accounts on several servers one has to remember many passwords. You can imagine a situation when someone forgets one of them. He/she remembers only that it consisted of words `x`, `y` and `z` as well as two digits: one at the very beginning and the other one at the end of the password.

Your task is to write a program which will generate all possible password on the basis of given dictionary and set of rules. For the example given above the dictionary contains three words: `x`, `y`, `z`, and the rule is given as `0#0` what stands for`SPMamp`<digit><word_from_the_dictionary><digit>&.

Input

First line contains a number of words in the dictionary (*n*). The words themselves are given in *n* consecutive lines. The next line contains number of rules (*m*). Similarly consecutive *m* lines contain rules. Each rule consists of characters `` `# ``' and `` `0 ``' given in arbitrary order. The character `` `# ``' stands for word from the dictionary whilst the character `` `0 ``' stands for a digit.

Input data may contain many sets of dictionaries with rules attached two them.

Output

For each set `dictionary + rules' you should output two hyphens followed by a linebreak and all matching passwords given in consecutive lines. Passwords should be sorted by rules what means that first all passwords matching the first rule and all words must be given, followed by passwords matching the second rule and all words, etc. Within set of passwords matching a word and a rule an ascending digit order must be preserved.

Assumptions: A number of words in the dictionary is greater than 0 and smaller or equal to 100 ($0 < n \leq 100$). Length of the word is greater than 0 and smaller than 256. A word may contain characters `A'..`Z',`a'..`z',`0'..`9'.A number of rules is smaller than 1000, and a rule is shorter that 256 characters. A character `0' may occur in the rule no more than 7 times, but it has to occur at least once. The character `#' is not mandatory meaning there can be no such characters in the rule.

Sample Input

```
2
root
2super
1
#0
1
admin
1
#0#
```

Sample Output

```
--
```

```
root0
root1
root2
root3
root4
root5
root6
root7
root8
root9
2super0
2super1
2super2
2super3
2super4
2super5
2super6
2super7
2super8
2super9
--
admin0admin
admin1admin
admin2admin
admin3admin
admin4admin
admin5admin
admin6admin
admin7admin
admin8admin
admin9admin
```

Miguel A. Revilla
2000-01-10

32. 11875 - Brick Game

Time limit: 1.000 seconds

H	Brick Game	
	Input	Standard Input
	Output	Standard Output

There is a village in Bangladesh, where brick game is very popular. Brick game is a team game. Each team consists of odd number of players. Number of players must be greater than 1 but cannot be greater than 10. Age of each player must be within 11 and 20. No two players can have the same age. There is a captain for each team. The communication gap between two players depends on their age difference, i.e. the communication gap is larger if the age difference is larger. Hence they select the captain of a team in such a way so that the number of players in the team who are younger than that captain is equal to the number of players who are older than that captain.

Ages of all members of the team are provided. You have to determine the age of the captain.

Input

Input starts with an integer **`T (T ≤ 100),`** the number of test cases.

Each of the next **T** lines will start with an integer **`N (1 < N < 11)`**, number of team members followed by **N** space separated integers representing ages of all of the members of a team. Each of these N integers will be between **`11`** and **`20`** (inclusive). Note that, ages will be given in strictly increasing order or strictly decreasing order. We will not mention which one is increasing and which one is decreasing, you have to be careful enough to handle both situations.

Output

For each test case, output one line in the format **`“Case x: a”`** (quotes for clarity), where **`x`** is the case number and a is the age of the captain.

Sample Input	**Sample Output**

`2`
`5 19 17 16 14 12`
`5 12 14 16 17 18`
	`Case 1: 16`
	`Case 2: 16`

Problemsetter: Md. Arifuzzaman Arif, Special Thanks: Shamim Hafiz

33. 374 - Big Mod

Time limit: 3.000 seconds

Big Mod

Calculate

$$R := B^P \bmod M$$

for large values of *B*, *P*, and *M* using an efficient algorithm. (That's right, this problem has a time dependency !!!.)

Input

Three integer values (in the order *B*, *P*, *M*) will be read one number per line. *B* and *P*are integers in the range 0 to 2147483647 inclusive. *M* is an integer in the range 1 to 46340 inclusive.

Output

The result of the computation. A single integer.

Sample Input

```
3
18132
17

17
1765
3

2374859
3029382
36123
```

Sample Output

```
13
2
13195
```

34. 612 - DNA Sorting

Time limit: 3.000 second.

DNA Sorting

One measure of ``unsortedness'' in a sequence is the number of pairs of entries that are out of order with respect to each other. For instance, in the letter sequence ``DAABEC'', this measure is 5, since D is greater than four letters to its right and E is greater than one letter to its right. This measure is called the number of inversions in the sequence. The sequence ``AACEDGG'' has only one inversion (E and D)--it is nearly sorted--while the sequence ``ZWQM'' has 6 inversions (it is as unsorted as can be--exactly the reverse of sorted).

You are responsible for cataloguing a sequence of DNA strings (sequences containing only the four letters A, C, G, and T). However, you want to catalog them, not in alphabetical order, but rather in order of ``sortedness'', from ``most sorted'' to ``least sorted''. All the strings are of the same length.

Input

The first line of the input is an integer M, then a blank line followed by M datasets. There is a blank line between datasets.

The first line of each dataset contains two integers: a positive integer *n* (0<n<=50) giving the length of the strings; and a positive integer *m* (0<m<=100) giving the number of strings. These are followed by *m* lines, each containing a string of length *n*.

Output

For each dataset, output the list of input strings, arranged from ``most sorted'' to ``least sorted''. If two or more strings are equally sorted, list them in the same order they are in the input file.

Print a blank line between consecutive test cases.

Sample Input

```
1

10 6
AACATGAAGG
TTTTGGCCAA
TTTGGCCAAA
GATCAGATTT
CCCGGGGGGA
ATCGATGCAT
```

Sample Output

```
CCCGGGGGGA
AACATGAAGG
GATCAGATTT
ATCGATGCAT
TTTTGGCCAA
TTTGGCCAAA
```

Miguel A. Revilla
1999-03-24

35. 623 - 500!

Time limit: 3.000 seconds

500!

In these days you can more and more often happen to see programs which perform some useful calculations being executed rather then trivial screen savers. Some of them check the system message queue and in case of finding it empty (for examples somebody is editing a file and stays idle for some time) execute its own algorithm.

As an examples we can give programs which calculate primary numbers.

One can also imagine a program which calculates a factorial of given numbers. In this case it is the time complexity of order $O(n)$ which makes troubles, but the memory requirements. Considering the fact that 500! gives 1135-digit number no standard, neither integer nor floating, data type is applicable here.

Your task is to write a programs which calculates a factorial of a given number.

Assumptions: Value of a number ``*n*'' which factorial should be calculated of does not exceed 1000 (although 500! is the name of the problem, 500! is a small limit).

Input

Any number of lines, each containing value ``*n*'' for which you should provide value of*n*!

Output

2 lines for each input case. First should contain value ``*n*" followed by character `!'. The second should contain calculated value *n*!.

Sample Input

```
10
30
50
100
```

Sample Output

```
10!
3628800
30!
265252859812191058636308480000000
50!
30414093201713378043612608166064768844377641568960512000000000000
100!
93326215443944152681699238856266700490715968264381621468592963895217599993229
91560894146397615651828625369792082722375825118521091686400000000000000000000
0000
```

Miguel A. Revilla
2000-01-10

36. 706 - LCD Display

Time limit: 3.000 seconds

LC-Display

A friend of you has just bought a new computer. Until now, the most powerful computer he ever used has been a pocket calculator. Now, looking at his new computer, he is a bit disappointed, because he liked the LC-display of his calculator so much. So you decide to write a program that displays numbers in an LC-display-like style on his computer.

Input

The input file contains several lines, one for each number to be displayed. Each line contains two integers *s*, *n* ($1 \leq s \leq 10, 0 \leq n \leq 99\,999\,999$), where *n* is the number to be displayed and *s* is the size in which it shall be displayed.

The input file will be terminated by a line containing two zeros. This line should not be processed.

Output

Output the numbers given in the input file in an LC-display-style using *s* ``-" signs for the horizontal segments and *s* ``|" signs for the vertical ones. Each digit occupies exactly *s*+2 columns and 2*s*+3 rows. (Be sure to fill all the white space occupied by the digits with blanks, also for the last digit.) There has to be exactly one column of blanks between two digits.

Output a blank line after each number. (You will find a sample of each digit in the sample output.)

Sample Input

```
2 12345
3 67890
0 0
```

Sample Output

```
      --   --        -- 
   |    |    | |  | |   
   |    |    | |  | |   
      --   --   --   -- 
   | |       |    |    |
   | |       |    |    |
      --   --        -- 

 ---   ---   ---   ---   --- 
|         | |   | |   | |   |
|         | |   | |   | |   |
|         | |   | |   | |   |
 ---         ---   ---       
|   |     | |   |     | |   |
|   |     | |   |     | |   |
|   |     | |   |     | |   |
 ---         ---   ---   --- 
```

Miguel A. Revilla
2000-02-09

37. 713 - Adding Reversed Numbers

Time limit: 3.000 seconds

Adding Reversed Numbers

The Antique Comedians of Malidinesia prefer comedies to tragedies. Unfortunately, most of the ancient plays are tragedies. Therefore the dramatic advisor of ACM has decided to transfigure some tragedies into comedies. Obviously, this work is very hard because the basic sense of the play must be kept intact, although all the things change to their opposites. For example the numbers: if any number appears in the tragedy, it must be converted to its reversed form before being accepted into the comedy play.

Reversed number is a number written in arabic numerals but the order of digits is reversed. The first digit becomes last and vice versa. For example, if the main hero had 1245 strawberries in the tragedy, he has 5421 of them now. Note that all the leading zeros are omitted. That means if the number ends with a zero, the zero is lost by reversing (e.g. 1200 gives 21). Also note that the reversed number never has any trailing zeros.

ACM needs to calculate with reversed numbers. Your task is to add two reversed numbers and output their reversed sum. Of course, the result is not unique because any particular number is a reversed form of several numbers (e.g. 21 could be 12, 120 or 1200 before reversing). Thus we must assume that no zeros were lost by reversing (e.g. assume that the original number was 12).

Input

The input consists of *N* cases. The first line of the input contains only positive integer *N*. Then follow the cases. Each case consists of exactly one line with two positive integers separated by space. These are the reversed numbers you are to add. Numbers will be at most 200 characters long.

Output

For each case, print exactly one line containing only one integer - the reversed sum of two reversed numbers. Omit any leading zeros in the output.

Sample Input

```
3
24 1
4358 754
305 794
```

Sample Output

```
34
1998
1
```

Miguel A. Revilla
2000-02-15

38. 12289 - One-Two-Three

Time limit: 1.000 seconds

One-Two-Three

Your little brother has just learnt to write one, two and three, in English. He has written a lot of those words in a paper, your task is to recognize them. Note that your little brother is only a child, so he may make small mistakes: for each word, there might be at most one wrong letter. The word length is always correct. It is guaranteed that each letter he wrote is in lower-case, and each word he wrote has a unique interpretation.

Input

The first line contains the number of words that your little brother has written. Each of the following lines contains a single word with all letters in lower-case. The words satisfy the constraints above: at most one letter might be wrong, but the word length is always correct. There will be at most 10 words in the input.

Output

For each test case, print the numerical value of the word.

Sample Input

```
3
owe
too
theee
```

Sample Output

```
1
2
3
```

The Seventh Hunan Collegiate Programming Contest
Problemsetter: Rujia Liu, Special Thanks: Yiming Li & Jane Alam Jan

39. 12243 - Flowers Flourish from France

Time limit: 3.000 seconds

Fiona has always loved poetry, and recently she discovered a fascinating poetical form.Tautograms are a special case of alliteration, which is the occurrence of the same letter at the beginning of adjacent words. In particular, a sentence is a tautogram if all of its words start with the same letter.

For instance, the following sentences are tautograms:

- Flowers Flourish from France
- Sam Simmonds speaks softly
- Peter pIckEd pePPers
- truly tautograms triumph

Fiona wants to dazzle her boyfriend with a romantic letter full of this kind of sentences. Please help Fiona to check if each sentence she wrote down is a tautogram or not.

Input

Each test case is given in a single line that contains a sentence. A sentence consists of a sequence of at most 50 words separated by single spaces. A word is a sequence of at most 20 contiguous uppercase and lowercase letters from the English alphabet. A word contains at least one letter and a sentence contains at least one word.

The last test case is followed by a line containing only a single character `*' (asterisk).

Output

For each test case output a single line containing an uppercase `Y' if the sentence is a tautogram, or an uppercase `N' otherwise.

Sample Input

```
Flowers Flourish from France
Sam Simmonds speaks softly
Peter pIckEd pePPers
truly tautograms triumph
this is NOT a tautogram
*
```

Sample Output

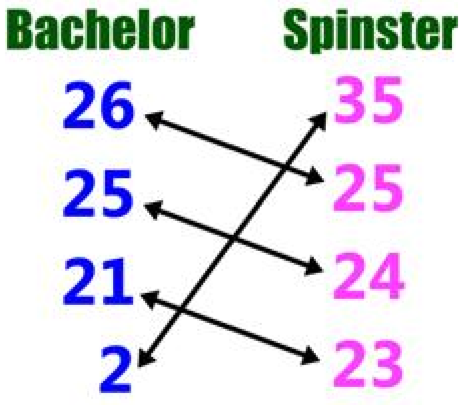
```
Y
Y
Y
Y
N
```

40. 12210 - A Match Making Problem

Time limit: 3.000 seconds

Match-making is a tough job and even then its long term success (A happy family) depends on two people who are often not involved in the match making process. But now, sites like facebook, twitter and communication devices/software like mobile phones, messengers have made the professional match makers jobless. So these angry and jobless match makers have gathered together to make the government pass a law in the parliament that will stop people from choosing their life partners. The law is stated below:

In a certain community, the most senior bachelor must marry a spinster (Female Bachelor) whose age is nearest to him. Then next senior bachelor will then marry a spinster whose age is nearest to him (of course if there is a tie, marrying anyone of them will do) excluding the spinster that has already got married. This process continues until there is no bachelor or spinster left. Of course a bachelor cannot marry two spinsters and a spinster cannot marry two bachelors. For example in a community there are four bachelors who are 21, 25, 26, 2 years old and four spinsters who are 26, 24, 25 and 35 years old. The diagram below shows the only possibility of marriage: (eg: The 26 year old bachelor marries the 25 years old spinster)



Now given the ages of the bachelors and the Spinsters in a community you will have to find the number of bachelors left, after all the marriages have taken place according to the law mentioned above. Also you have to report the age of the youngest bachelor left in the community if there is one.

Input

The input file contains at most 25 sets of inputs. The description of each set is given below:

The first line of each set contains two integers B (0<B<10000) and S (0<S<10000) which denotes the total number of bachelors and spinsters in the community respectively. Each of the next B lines contains one integer between 2 and 60 (inclusive) which denotes the age of one bachelor in the community. Each of the next S lines contains one integer between 2 and 60 (inclusive) which denotes the age of one spinster in the community. For simplicity you don’t need to worry about getting married at a very small age in this problem. That means unmarried people of all ages are valid bachelor or spinster.

Input is terminated by a line containing two zeroes.

Output

For each set of input produce one line of output. This line contains the serial of output followed by one or two integers. The first integer denotes the number of bachelors left in the community after all potential marriages have been completed. If this integer is not zero then print a second integer which denotes the age of the youngest bachelor left in the community after all possible marriages have been completed. Look at the output for sample input for details.

Sample Input

```
4 4

26

25

2

21

35

25

23

24

1 2

20

30

40

4 2

5

5

10

15
```

Output for Sample Input

```
Case 1: 0

Case 2: 0

Case 3: 2 5
```

20
 	
18
 	
0 0	

41. 11953 - Battleships

Time limit: 1.000 seconds

C — Battleships

Time Limit: 1 sec
Memory Limit: 32 MB

Battleships game is a pen and paper game that was invented by Clifford Von Wickler in the early 1900s. In this game each player uses two *N* x *N* grids. One to arrange his ships and record the shots of the opponent. On the other grid the player records his own shots. Ships in battleship game can vary in size from 1 x 1 to 1 x *N*/2 and can be placed both vertically and horizontally. When all of the ship's cells have been hit, the ship is considered sunk, otherwise it is still "alive". Beside this, there can be more than one ship of each size, however none of two ships can overlap or touch.

In this problem you will be given the placement of ships on the player's grid. You will have to calculate the number of ships that the player still owns.

INPUT

There is a number of tests *T* ($T \le 100$) on the first line. Each test case contains a positive number *N* ($N \le 100$) — grid size. Next *N* lines contain *N* characters each, describing the playing grid. Character `"."` stands for an empty cell, `"x"` for a cell containing a ship or its part and `"@"` for already hit part of a ship.

OUTPUT

For each test case output a single line `"Case T: N"`. Where *T* is the test case number (starting from 1) and *N* is the number of still "alive" ships.

SAMPLE INPUT

```
2
4
```

```
x...
..x.
@.@.
....
2
..
x.
```

SAMPLE OUTPUT

```
Case 1: 2
Case 2: 1
```

Problem by: Aleksej Viktorchik; Leonid Sislo
Huge Easy Contest #2

42. 10954 - Add All

Time limit: 3.000 seconds

Problem F

Add All

Input: standard input
Output: standard output

Yup!! The problem name reflects your task; just add a set of numbers. But you may feel yourselves condescended, to write a C/C++ program just to add a set of numbers. Such a problem will simply question your erudition. So, let's add some flavor of ingenuity to it.

Addition operation requires cost now, and the cost is the summation of those two to be added. So, to add **1** and **10**, you need a cost of **11**. If you want to add **1**, **2**and **3**. There are several ways –

1 + 2 = 3, cost = 3	1 + 3 = 4, cost = 4	2 + 3 = 5, cost = 5

3 + 3 = 6, cost = 6 Total = 9	2 + 4 = 6, cost = 6 Total = 10	1 + 5 = 6, cost = 6 Total = 11

I hope you have understood already your mission, to add a set of integers so that the cost is minimal.

Input

Each test case will start with a positive number, **N (2 ≤ N ≤ 5000)** followed by **N**positive integers (all are less than **100000**). Input is terminated by a case where the value of **N** is zero. This case should not be processed.

Output

For each case print the minimum total cost of addition in a single line.

Sample Input	Output for Sample Input

3
 	
1 2 3
 	
4
 	
1 2 3 4
 	
0
	9
	
	19

Problem setter: Md. Kamruzzaman, EPS

43. 10363 - Tic Tac Toe

Time limit: 3.000 seconds

Problem B: Tic Tac Toe

Tic Tac Toe is a child's game played on a 3 by 3 grid. One player, X, starts by placing an X at an unoccupied grid position. Then the other player, O, places an O at an unoccupied grid position. Play alternates between X and O until the grid is filled or one player's symbols occupy an entire line (vertical, horizontal, or diagonal) in the grid.

We will denote the initial empty Tic Tac Toe grid with nine dots. Whenever X or O plays we fill in an X or an O in the appropriate position. The example below illustrates each grid configuration from the beginning to the end of a game in which X wins.

```
...  X..  X.O  X.O  X.O  X.O  X.O  X.O
...  ...  ...  ...  .O.  .O.  OO.  OO.
...  ...  ...  ..X  ..X  X.X  X.X  XXX
```

Your job is to read a grid and to determine whether or not it could possibly be part of a valid Tic Tac Toe game. That is, is there a series of plays that can yield this grid somewhere between the start and end of the game?

The first line of input contains N, the number of test cases. 4N-1 lines follow, specifying N grid configurations separated by empty lines. For each case print "yes" or "no" on a line by itself, indicating whether or not the configuration could be part of a Tic Tac Toe game.

Sample Input

```
2
X.O
OO.
XXX

O.X
XX.
OOO
```

Output for Sample Input

```
yes
no
```

44. 492 - Pig-Latin

Time limit: 3.000 seconds

Pig-Latin

You have decided that PGP encryptation is not strong enough for your email. You have decided to supplement it by first converting your clear text letter into Pig Latin before encrypting it with PGP.

Input and Output

You are to write a program that will take in an arbitrary number of lines of text and output it in Pig Latin. Each line of text will contain one or more words. A ``word'' is defined as a consecutive sequence of letters (upper and/or lower case). Words should be converted to Pig Latin according to the following rules (non-words should be output exactly as they appear in the input):

1. Words that begin with a vowel (`a`, `e`, `i`, `o`, or `u`, and the capital versions of these) should just have the string ``ay'' (not including the quotes) appended to it. For example, ``apple'' becomes ``appleay''.
2. Words that begin with a consonant (any letter than is not `A`, `a`, `E`, `e`, `I`, `i`, `O`, `o`, `U` or `u`) should have the first consonant removed and appended to the end of the word, and then appending ``ay'' as well. For example, ``hello'' becomes ``ellohay''.
3. Do not change the case of any letter.

Sample Input

```
This is the input.
```

Sample Output

```
hisTay isay hetay inputay.
```

45. 10070 - Leap Year or Not Leap Year and …

Time limit: 3.000 seconds

Problem A

Leap Year or Not Leap Year and …

Input: standard input

Output: standard output

The ancient race of Gulamatu is very advanced in their year calculation scheme. They understand what leap year is (A year that is divisible by 4 and not divisible by 100 with the exception that years that are divisible by 400 are also leap year.) and they have also similar festival years. One is the Huluculu festival (happens on years divisible by 15) and the Bulukulu festival (Happens on years divisible by 55 provided that is also a leap year). Given an year you will have to state what properties these years have. If the year is not leap year nor festival year, then print the line 'This is an ordinary year.' The order of printing (if present) the properties is leapyear-->huluculu-->bulukulu.

Input

Input will contain several years as input. Each year will be in separate lines. Input is terminated by end of file. All the years will not be less than 2000 (to avoid the earlier different rules for leap years). Please don't assume anything else.

Output

For each input, output the different properties of the years in different lines according to previous description and sample output. A blank line should separate the output for each line of input. Note that there are four different properties.

Sample Input

```
2000
3600
4515
2001
```

Sample Output

```
This is leap year.

This is leap year.
This is huluculu festival year.

This is huluculu festival year.

This is an ordinary year.
```

Shahriar Manzoor

46. 10474 - Where is the Marble?

Time limit: 3.000 seconds

Where is the Marble?

Raju and Meena love to play with Marbles. They have got a lot of marbles with numbers written on them. At the beginning, Raju would place the marbles one after another in ascending order of the numbers written on them. Then Meena would ask Raju to find the first marble with a certain number. She would count 1...2...3. Raju gets one point for correct answer, and Meena gets the point if Raju fails. After some fixed number of trials the game ends and the player with maximum points wins. Today it's your chance to play as Raju. Being the smart kid, you'd be taking the favor of a computer. But don't underestimate Meena, she had written a program to keep track how much time you're taking to give all the answers. So now you have to write a program, which will help you in your role as Raju.

Input

There can be multiple test cases. Total no of test cases is less than 65. Each test case consists begins with 2 integers: ***N*** the number of marbles and ***Q*** the number of queries Mina would make. The next ***N*** lines would contain the numbers written on the ***N*** marbles. These marble numbers will not come in any particular order. Following ***Q*** lines will have ***Q*** queries. Be assured, none of the input numbers are greater than 10000 and none of them are negative.

Input is terminated by a test case where ***N*** = 0 and ***Q*** = 0.

Output

For each test case output the serial number of the case.

For each of the queries, print one line of output. The format of this line will depend upon whether or not the query number is written upon any of the marbles. The two different formats are described below:

- `` `x found at y' ``, if the first marble with number ***x*** was found at position ***y***. Positions are numbered 1, 2,..., ***N***.
- `` `x not found' ``, if the marble with number ***x*** is not present.

Look at the output for sample input for details.

Sample Input

```
4 1
2
3
5
1
5
5 2
1
3
3
3
1
2
3
0 0
```

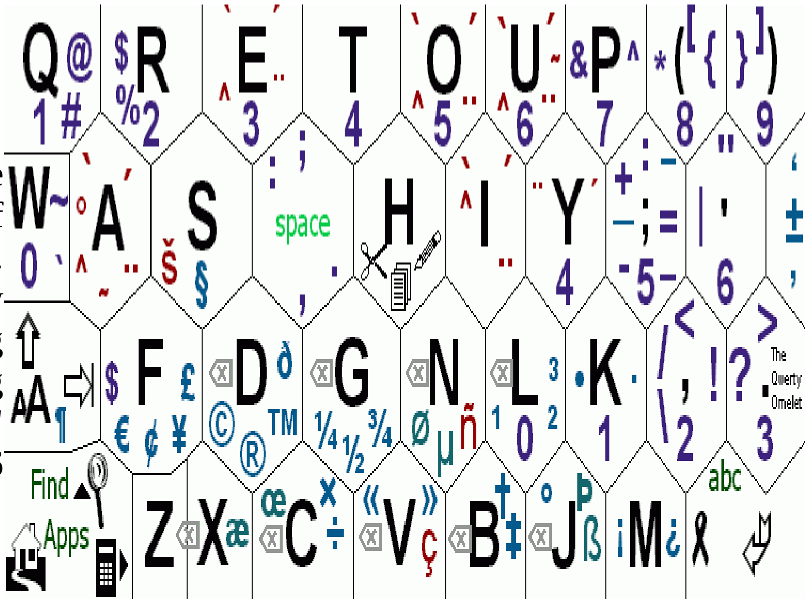
Sample Output

```
CASE# 1:
5 found at 4
CASE# 2:
2 not found
3 found at 3
```

Problem-setter: Monirul Hasan Tomal, Southeast University

47. 11577 - Letter Frequency

Time limit: 1.000 seconds



Problem A: Letter Frequency

In this problem we are interested in the frequency of letters in a given line of text. Specifically, we want to know the most frequently occurring letter(s) in the text, ignoring case (to be clear, "letters" refers precisely to the 26 letters of the alphabet).

Input begins with the number of test cases on its own line. Each test case consists of a single line of text. The line may contain non-letter characters, but is guaranteed to contain at least one letter and less than 200 characters in total.

For each test case, output a line containing the most frequently occurring letter(s) from the text in lowercase (if there are ties, output all such letters in alphabetical order).

Sample input

```
1
Computers account for only 5% of the country's commercial electricity consumption.
```

Sample output

```
co
```

Sumudu Fernando

48. 11636 - Hello World!

Time limit: 1.000 seconds

Problem A

Hello World!

Input: Standard Input

Output: Standard Output

When you first made the computer to print the sentence “`Hello World!`”, you felt so happy, not knowing how complex and interesting the world of programming and algorithm will turn out to be. Then you did not know anything about loops, so to print 7 lines of “Hello World!”, you just had to copy and paste some lines. If you were intelligent enough, you could make a code that prints “Hello World!” 7 times, using just 3 paste commands. Note that we are not interested about the number of copy commands required. A simple program that prints “Hello World!” is shown in Figure

1. By copying the single print statement and pasting it we get a program that prints two “Hello World!” lines. Then copying these two print statements and pasting them, we get a program that prints four “Hello World!” lines. Then copying three of these four statements and pasting them we can get a program that prints seven “Hello World!” lines (Figure 4). So three pastes commands are needed in total and Of course you are not allowed to delete any line after pasting. Given the number of “Hello World!” lines you need to print, you will have to find out the minimum number of pastes required to make that program from the origin program shown in Figure 1.

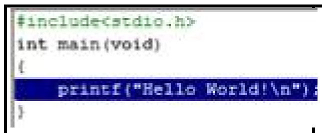
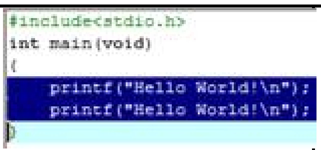
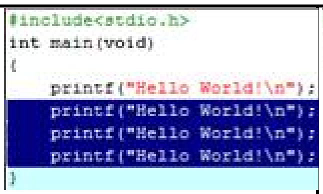
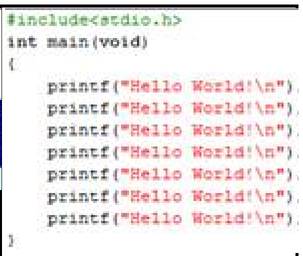
			

Figure 1	Figure 2	Figure3	Figure 4

```
#include<stdio.h>
int main(void)
{
    printf("Hello World!\n");
}
```

```
#include<stdio.h>
int main(void)
{
    printf("Hello World!\n");
    printf("Hello World!\n");
}
```

```
#include<stdio.h>
int main(void)
{
    printf("Hello World!\n");
    printf("Hello World!\n");
    printf("Hello World!\n");
    printf("Hello World!\n");
}
```

```
#include<stdio.h>
int main(void)
{
    printf("Hello World!\n")
    printf("Hello World!\n")
    printf("Hello World!\n")
    printf("Hello World!\n")
    printf("Hello World!\n")
    printf("Hello World!\n")
    printf("Hello World!\n")
}
```

Input

The input file can contain up to 2000 lines of inputs. Each line contains an integer N (0<N<10001) that denotes the number of “Hello World!” lines are required to be printed.

Input is terminated by a line containing a negative integer.

Output

For each line of input except the last one, produce one line of output of the form “Case X: Y” where X is the serial of output and Y denotes the minimum number of paste commands required to make a program that prints N lines of “Hello World!”.

Sample Input	Output for Sample Input
2	Case 1: 1
10	Case 2: 4
-1	

49. 10783 - Odd Sum

Time limit: 3.000 seconds

Odd Sum

Given a range [***a***, ***b***], you are to find the summation of all the odd integers in this range. For example, the summation of all the odd integers in the range [3, 9] is 3 + 5 + 7 + 9 = 24.

Input

There can be at multiple test cases. The first line of input gives you the number of test cases, ***T*** ($1 \le T \le 100$). Then T test cases follow. Each test case consists of 2 integers ***a*** and ***b*** ($0 \le a \le b \le 100$) in two separate lines.

Output

For each test case you are to print one line of output - the serial number of the test case followed by the summation of the odd integers in the range [***a***, ***b***].

Sample Input

```
2
1
5
3
5
```

Sample Output

```
Case 1: 9
Case 2: 8
```

Miguel Revilla 2004-12-02

50. 483 - Word Scramble

Time limit: 3.000 seconds

Word Scramble

Write a program that will reverse the letters in each of a sequence of words while preserving the order of the words themselves.

Input

The input file will consist of several lines of several words. Words are contiguous stretches of printable characters delimited by white space.

Output

The output will consist of the same lines and words as the input file. However, the letters within each word must be reversed.

Sample Input

```
I love you.
You love me.
We're a happy family.
```

Sample Output

```
I evol .uoy
uoY evol .em
er'eW a yppah .ylimaf
```