Ambiguous Watch

You have a watch with only an hour hand and a minute hand. The watch is a round 12-hour watch. Both hands move continuously. Both hands have the same length and are indistinguishable, so at certain moments, the time displayed is ambiguous. A time is considered ambiguous if there exists a different time at which the hands appear to have the same positions. For example, the time moment on the picture is ambiguous. At this moment, you would not be able to tell if it was a little past 00:05, or a little past 01:00.

You are given two times in the form "HH:MM" (quotes for clarity), where 00 <= HH < 12, and 00 <= MM < 60. Print the number of ambiguous moments between ***startTime*** and ***finishTime***, inclusive. The times represented by ***startTime*** and ***finishTime*** are in the same half of the day.

Notes:

- startTime and finishTime will have the form "HH:MM" (quotes for clarity), where 00 <= HH < 12, 00 <= MM < 60.
- startTime must be earlier than or the same as finishTime (in the same half of the day).

Input Specification

The input will contain several test cases, each test cases will be in a single line, with ***startTime*** and ***finishTime*** separated by a space, ***startTime*** will always be first in the line.

Output Specification

Print one line per test case with the answer, follow the format below

Input Example

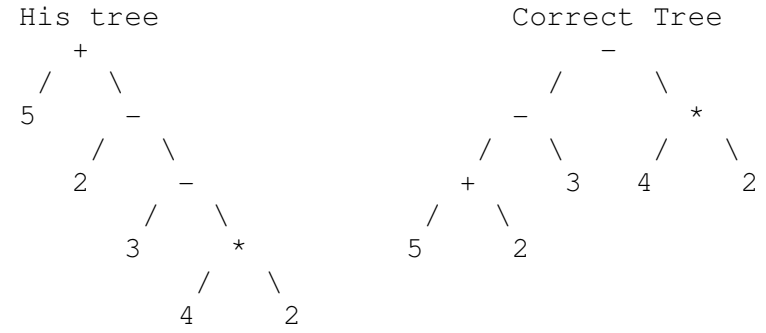
```
00:00 00:00
00:05 00:06
00:00 01:00
00:00 11:59
03:21 10:54
00:00 00:22
```

Output Example

```
0
1
11
132
84
4
```

BadParser

You and a partner have teamed up for a coding project. He is writing the front-end of an arithmetic expression parser, and you are writing the back-end. The expressions are pretty simple, with normal arithmetic operations and no parentheses. As usual, he stayed up too late and made a terrible oversight. His parser spits out an expression tree where the precedence and associativity of the operators may be ignored. For example, let's say his program is given the expression "5+2-3-4*2". Each operator is supposed to be left associative, but his program could spit out the wrong tree:



The expressions should be interpreted as follows:

1) As usual, the order of operations gives * and / highest precedence whereas + and - have lowest. * and / have equal precedence. In addition, + and - have equal precedence.
2) Amongst operations of equal precedence, process the leftmost operation first.

Adhering to these rules, the input above would be processed to produce the Correct Tree. In such a tree, lower nodes are processed before higher nodes. The value of a number node is the number itself. The value of an operation node is that operation applied to the values of its left and right subnodes (the value of the left subnode belongs on the left side of the operation). The value of the tree is the value of the top node (called the root node). Unfortunately, your partner's front-end may have violated rules 1 and 2 numerous times. Luckily the ordering of his tree is not messed up. This means that an inorder traversal of the tree beginning with the root will produce the original expression. Inorder traversal pseudocode follows:

```
InorderTraverse(node) {
      if (left subtree of node exists)
            InorderTraverse(root of left subtree)
      Print the contents of node
      if (right subtree of node exists)
            InorderTraverse(root of right subtree)
}
```

Your program will take a sequence of lines; ***badTree*** describing his tree, and will print an integer value which is the correct value of the expression he parsed. All operations are integer operations so division truncates results. For example 5/3=1, and -5/3 = -1.

Notes:

- Each element of ***badTree*** will be in one of two forms (quotes for clarity):
  - "op X Y" : op is one of *,/,+,-. X,Y are integers referencing other elements of ***badTree*** (0-indexed). X refers to the node's left child and Y refers to the node's right child.
  - "N" : N is a non-negative integer with no extra leading zeros.
- ***badTree*** will contain between 1 and 50 elements inclusive.

- Each element of ***badTree*** will be in one of the following forms (quotes for clarity):
  - "op X Y" where X and Y are integers, with no extra leading zeros, between 0 and M-1 inclusive. op must be *,+,-, or /. Here M denotes the number of elements in badTree.
  - "N" where N is a nonnegative integer with no extra leading zeros between 0 and 1000 inclusive
- The elements of ***badTree*** will describe a single tree, with element 0 acting as root.
- The value to print, and the value of any subtree of the correct tree will all be between -100000 and 100000 inclusive.
- The computation of the value to print, and the value of any subtree of the correct tree will not require division by 0.

Input Specification
The input will contain several test cases, each test cases will be in a single line, each element in the ***badTree*** sequence will be separated by a comma (,).

Output Specification
Print one line per test case with the answer, follow the format below

Input Example
```
+ 1 2,5,- 3 4,2,- 5 6,3,* 7 8,4,2
- 1 2,5,- 3 4,2,- 5 6,3,* 7 8,4,2
* 1 2,5,- 3 4,2,- 5 6,3,* 7 8,4,2
99
* 1 2,+ 3 4,+ 5 6,200,200,200,200
* 1 2,2,* 3 4,2,* 5 6,2,* 7 8,2,* 9 10,2,- 11 12,2,- 13 14,2,- 15 16,2,2
/ 1 2,/ 3 4,/ 5 6,4,1,4,2
```

Output Example
```
-4
-8
-1
99
40400
58
0
```

Cable Donation

A local charity is asking for donations of Ethernet cable. You realize that you probably have a lot of extra cable in your house, and make the decision that you will donate as much cable as you can spare.

You will be given a sequence of strings; ***lengths*** indicating the length (in meters) of cables between each pair of rooms in your house. You wish to keep only enough cable so that every pair of rooms in your house is connected by some chain of cables, of any length. The jth character of lengths[i] gives the length of the cable between rooms i and j in your house. A value of '0' (zero) indicates no cable, values of 'a' through 'z' indicate lengths of 1 through 26, and values of 'A' through 'Z' indicate lengths of 27 through 52.

If both the jth character of lengths[i] and the ith character of lengths[j] are greater than 0, this means that you have two cables connecting rooms i and j, and you can certainly donate at least one of them. If the ith character of lengths[i] is greater than 0, this indicates unused cable in room i, which you can donate without affecting your home network in any way.

You are not to rearrange any cables in your house; you are only to remove unnecessary ones. Find and print the maximum total length of cables (in meters) that you can donate. If any pair of rooms is not initially connected by some path, print -1.

Notes:

- ***lengths*** will contain between 1 and 50 elements, inclusive.
- The length of each element of lengths will be equal to the number of elements in lengths.
- Each character in lengths will be a letter ('a'-'z', 'A'-'Z') or '0' (zero).

Input Specification

The input will contain several test cases, each test cases will be given in a sequence of lines

Output Specification

Print one line per test case with the answer, follow the format below

Input Example

```
abc
def
ghi
a0
0b
0X00
00Y0
0000
00Z0
0000
b000
0b00
00b0
Az
aZ
0top
c0od
er0o
pen0
c
```

Output Example

```
40
-1
0
0
105
134
3
```

Date Format

In the US, dates are usually written starting with the month, but in Europe, they are usually written starting with the day. So, January 16 will be written as "01/16" in the US and as "16/01" in Europe.

You have a list of dates for the next year and it is known that the given dates are listed in strictly increasing order. Unfortunately, the list was populated by different people and it can contain dates in both formats. You want to convert all dates into the US format.

You will be given several lines; the ***dateList***. First, you should concatenate all elements of ***dateList*** and consider it as one line (or String). The conjoint ***dateList*** will contain a space-separated list of the dates. Each date will be in the form "XX/XX" (quotes for clarity), where each X is a digit. Convert the dates (without changing the order of the list) so that each date is in the US format and the list is in strictly increasing order. Note that in the original list, the format in which certain dates were written might be ambiguous. You may interpret those dates as being in either format as long as the final list is in strictly increasing order. Print the result as a single Line in the same format as the original. If there are several solutions possible print the one that comes first lexicographically. If it is impossible to obtain a strictly increasing list of dates, print an empty Line.

Notes:

- 2008,is a leap year. So, February 29 is a valid date.
- ***dateList*** will contain between 1 and 50 elements, inclusive.
- Each element of ***dateList*** will contain between 1 and 50 characters, inclusive.
- The conjoint ***dateList*** will contain a single space separated list of dates without leading or trailing spaces.
- Each date in ***dateList*** will be in the form "XX/XX" (quotes for clarity), where each X is a digit.
- Each date in ***dateList*** will represent a valid date in either US format or European format.

Input Specification

The input will contain several test cases, each test cases will be in several lines, the first line will contain an Integer ***N***, the numbers of lines that follow, the next ***N*** line will contain the ***dateList***.

Output Specification

Print one line per test case with the answer, follow the format below

Input Example

```
1
16/01
1
02/01 08/02 08/02 21/09 06/11
1
08/02 08/02 03/04
4
2
9/02
 08/
03 01/08
```

Output Example

```
01/16
01/02 02/08 08/02 09/21 11/06

02/29 03/08 08/01
```

Education

Even students who hate math find one calculation very useful -- what is the lowest score I can get on the last test and pull out a certain grade? Let's write a program to help them minimize their education.
We will assume that an average score of 90 or higher is rewarded with an A grade, 80 or higher (but less than 90) is a B, 70 or higher (but less than 80) is a C, 60 or higher (but less than 70) is a D. All test scores are integers between 0 and 100 inclusive and the average is NOT rounded -- for example an average of 89.99 does NOT get you an A.

You should write a program that take the desire grade and a sequence of integers ***tests*** containing the scores on all but the final test. Print the lowest possible test score for the final test that will earn at least the desired grade. If even a perfect score won't get the desired grade, Print -1.

The desired grade will be given as a String of length 1, either "A", "B", "C", or "D".

Input Specification
The input will contain several test cases, each test cases will be in two lines:

- A line with the desire grade, with no leading or trailing spaces
- A line with a sequence of integers separated by single spaces, all between 0 and 100 inclusive; the given scores. There will not be leading or trailing spaces on this line.

Output Specification
Print one line per test case with the answer, follow the format below

Input Example

```
A
0 70
D
100 100 100 100 100 100
B
80 80 80 73
B
80 80 80 73 79
A
80
```

Output Example

```
-1
0
87
88
100
```

Factorial GCD

The greatest common divisor (GCD) of two positive integers a and b is the largest integer that evenly divides both a and b. In this problem, you will find the GCD of a positive integer and the factorial of a non-negative integer.

You must find the GCD of a! (the factorial of a) and b.

Notes:

- Assume 0! = 1.
- a will be between 0 and 2147483647, inclusive.
- b will be between 1 and 2147483647, inclusive.

Input Specification

The input will contain several test cases, each test cases will be in a single line, it will contain ***a*** and ***b,*** separated by a space and without leading or trailing spaces.

Output Specification

Print one line per test case with the answer, follow the format below.

Input Example

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

Output Example

```
1
2
3
```

Game Of Life

Cat Taro and Rabbit Hanako invented a new variation of "Game Of Life". N cells are arranged around a circle. The cells are numbered from 0 to N-1. For each i between 0 and N-2, inclusive, the i-th cell and the (i+1)-th cell are adjacent to each other. The (N-1)-th cell and the 0-th cell are adjacent to each other. Each cell has exactly two adjacent cells. Each cell has a state: "live" or "die".

Taro and Hanako can decide the states of the cells at time 0. For time t > 0, the states are determined as follows: Consider three cells: the i-th cell and the two cells that are adjacent to the i-th cell.

- If at least two of the three cells are "live" at time t-1, the state of the i-th cell at time t will be "live".
- If at least two of the three cells are "die" at time t-1, the state of the i-th cell at time t will be "die".

You are given the initial configuration. The number of cells in the game (N) is equal to the number of characters in the given sequence. The i-th character of init represents the state they assign to the i-th cell at time 0. '1' means "live" and '0' means "die". Print a string that describes the states at time ***T*** using the same encoding. Each character in the initial configuration will be '0' (zero) or '1' (one).

Input Specification
The input will contain several test cases, each test cases will be in a single a line with the ***initial configuration*** and ***T***.

Output Specification
Print one line per test case with the answer, follow the format below

Input Example
```
01010 2
```

Output Example
```
00000
```

Hangman

You are playing a game of Hangman with your brother. He begins by choosing a word, and he writes down one blank for each letter in the word. You then repeatedly guess letters that you think could be in the word. Every time you guess a letter, your brother indicates every occurrence of that letter in the word by replacing blanks with the corresponding letter. For example, suppose your brother chooses the word, "NINJA".

He begins by writing down 5 blanks, indicated here by dashes: "-----".
Suppose you first guess the letter 'A'. Then, your brother would reveal the one instance of 'A' in the word: "----A".
Suppose you next guess the letter 'N'. Then, your brother would reveal both instances of 'N' in the word: "N-N-A".
Suppose you next guess the letter 'E'. Then, your brother would reveal nothing new since 'E' is not in the word: "N-N-A".

You will be given a String ***feedback*** and a sequence of ***words***. Since your brother is fairly predictable, you have determined that the word he is thinking of is an element of words. You also have the information he has given you after some number of guesses (possibly 0), given in feedback. As above, blanks are represented by dashes ('-'), and revealed letters are represented by capital letters ('A'-'Z'). You now wish to guess the word that your brother is thinking of. If there is exactly one element of ***words*** that is consistent with feedback, you should print that element. Otherwise, you should print an empty line.

Notes:
feedback will contain between 1 and 50 characters inclusive.
Each character in ***feedback*** will be either a capital letter ('A'-'Z') or a dash ('-').
words will contain between 1 and 50 elements inclusive.
Each element in ***words*** will contain between 1 and 50 characters inclusive.
No two elements in ***words*** will be equal.
Each character in each element of ***words*** will be a capital letter ('A'-'Z').

Input Specification
The input will contain several test cases, each test cases will be in two lines, a line with the ***feedback***, and another with the given ***words.***

Output Specification
Print one line per test case with the answer, follow the format below

Input Example
```
N-N-A
NINJA NINJAS FLIPS OUT FRISBEE
B--B--
BRINGS BARBED BUBBLE
---------
MONKEY FORCE IS GAINING STRENGTH
-AAA--
CAAABB BAAACC
```

Output Example
```
NINJA
BARBED
```

Image Compress

Your task is to convert a black-and-white image into the compressed format described below. You should print the shortest possible encoding for the image. If more than one encoding achieves the minimum length, print the one that comes first alphabetically.

The encoding format is based on the idea of recursively decomposing an image into subimages until each subimage is composed of a single color. For example, the image

BBBWWW
BBBWWW

might be decomposed into two 2-by-3 subimages:

BBB WWW
BBB WWW

The black subimage could then be encoded as 'B' and the white subimage could be encoded as 'W'. The entire decomposition would be encoded as "LBW". An image can be decomposed in four different ways, each indicated by a single character:

- 'L' indicates that the image is decomposed into its left half and its right half (if the image contains an odd number of columns, the center column is considered part of the left half).
- 'U' indicates that the image is decomposed into its upper half and its lower half (if the image contains an odd number of rows, the center row is considered part of the upper half).
- 'C' indicates that the image is decomposed into even columns and odd columns (the leftmost column is considered column 0, and is therefore even).
- 'R' indicates that the image is decomposed into even rows and odd rows (the topmost row is considered row 0, and is therefore even).

The letters 'B' and 'W' indicate that the image is completely black or completely white, respectively. If the image contains a mix of black and white, then it is decomposed in one of the four ways. The image is encoded as the single letter for the decomposition pattern, followed by the encoding of the left/upper/even subimage, followed by the encoding of the right/lower/odd subimage. An image that contains a single column will never be decomposed using 'L' or 'C', and an image that contains a single row will never be decomposed using 'U' or 'R'.

For example, the image

BWB
WWW

could be encoded in a minimum of 5 characters in any of the following ways: "CRBWW", "CUBWW", "RCBWW", or "UCBWW". Of these, "CRBWW" is the first alphabetically, so it is the preferred answer. The 'C' indicates that the original image is decomposed into the two subimages

BB W
WW W

The 2-by-2 subimage is then encoded as "RBW" and the all-white 2-by-1 subimage is encoded simply as 'W'.

The image will be given as a rectangular Matrix of characters; ***image*** containing the characters 'B' and 'W'. Each element of image represents a row of the image. For example, see the image

```
BBBWWW
WWWBBB
BBWWBB
```

Notes:

Decompressing a compressed image requires knowledge of the original image's size, as well as the information in the format described here. Do not be concerned that the size is not encoded in the compressed format.

image contains between 1 and 30 elements, inclusive.

Each element of ***image*** contains between 1 and 30 characters, inclusive.

Each element of ***image*** contains the same number of characters.

Every character in ***image*** is a 'B' or a 'W'.

Input Specification
The input will contain several test cases, each test cases will start with a line containing ***R*** and ***C***, the numbers of rows and columns of the ***image*** that follows, then R lines; the ***image*** itself.

Output Specification
Print one line per test case with the answer, follow the format below

Input Example

```
2 6
BBBWWW
BBBWWW
2 3
BWB
WWW
5 8
BWBWBWBW
WBWBWBWB
BWBWBWBW
WBWBWBWB
BWBWBWBW
30 30
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
BBBBBWWWWWBBBBBWWWWWBBBBBWWWWW
3 6
BBBWWW
WWWBBB
```

```
BBWWBB
```

Output Example

```
LBW
CRBWW
CRBWRWB
LLLBCCBWWLLWBBLLWCCWBBLLBWW
RRLBWCCBWCBWLWB
```

Jewelry

You have been given a list of jewelry items that must be split amongst two people: Frank and Bob. Frank likes very expensive jewelry. Bob doesn't care how expensive the jewelry is, as long as he gets a lot of jewelry. Based on these criteria you have devised the following policy:
1) Each piece of jewelry given to Frank must be valued greater than or equal to each piece of jewelry given to Bob. In other words, Frank's least expensive piece of jewelry must be valued greater than or equal to Bob's most expensive piece of jewelry.
2) The total value of the jewelry given to Frank must exactly equal the total value of the jewelry given to Bob.
3) There can be pieces of jewelry given to neither Bob nor Frank.
4) Frank and Bob must each get at least 1 piece of jewelry.
Given the value of each piece (between 2 and 30, and each value between 1 and 1000), you will determine the number of different ways you can allocate the jewelry to Bob and Frank following the above policy. For example:

```
      values = {1,2,5,3,4,5}
Valid allocations are:
  Bob                    Frank
  1,2                      3
  1,3                      4
  1,4                      5  (first 5)
  1,4                      5  (second 5)
  2,3                      5  (first 5)
  2,3                      5  (second 5)
   5  (first 5)            5  (second 5)
   5  (second 5)           5  (first 5)
1,2,3,4                   5,5
```

Note that each '5' is a different piece of jewelry and needs to be accounted for separately. There are 9 legal ways of allocating the jewelry to Bob and Frank given the policy, so you should print 9.

Input Specification
The input will contain several test cases, each test cases will be in a single line.

Output Specification
Print one line per test case with the answer, follow the format below

Input Example

```
1 2 5 3 4 5
1 2 3 4 5 6
1 2 3 4 5
7 7 8 9 10 11 1 2 2 3 4 5 6
```

Output Example

```
9
7
4
607
```

KiloManX

The KiloMan series has always had a consistent pattern: you start off with one (rather weak) default weapon, and then defeat some number of bosses. When you defeat a boss, you then acquire his weapon, which can then be used against other bosses, and so on. Usually, each boss is weak against some weapon acquired from another boss, so there is a recommended order in which to tackle the bosses. You have been playing for a while and wonder exactly how efficient you can get at playing the game. Your metric for efficiency is the total number of weapon shots fired to defeat all of the bosses.

You have a chart in front of you detailing how much damage each weapon does to each boss per shot, and you know how much health each boss has. When a boss's health is reduced to 0 or less, he is defeated. You start off only with the Kilo Buster, which does 1 damage per shot to any boss. The chart is represented as a sequence of strings, with the ith element containing N one-digit numbers ('0'-'9'), detailing the damage done to bosses 0, 1, 2, ..., N-1 by the weapon obtained from boss i, and the health is represented as a sequence of integers values, with the ith element representing the amount of health that boss i has.

Given a sequence of strings; ***damageChart,*** representing all the weapon damages, and a sequence of integers values; ***bossHealth,*** showing how much health each boss has, you should print an integer value which is the least number of shots that need to be fired to defeat all of the bosses.

Notes:

- ***damageChart*** will contain between 1 and 15 elements, inclusive.
- each element of ***damageChart*** will be of the same length, which will be the same as the number of elements in ***damageChart***.
- each element of ***damageChart*** will contain only the characters '0'-'9'.
- ***bossHealth*** will contain between 1 and 15 elements, inclusive.
- ***damageChart*** and ***bossHealth*** will contain the same number of elements.
- each element in ***bossHealth*** will be between 1 and 1000000, inclusive.

Input Specification

The input will contain several test cases, each test cases will consist of two lines, a line containing the sequence of strings ***damageChart***, and another line containing the sequence of integers values ***bossHealth***.

Output Specification

Print one line per test case with the answer, follow the format below

Note about the input example 4: there are 15 values on the damageChart sequence, those values are in bold.

Input Example

```
070 500 140
50 150 150
1542 7935 1139 8882
50 150 150 150
07 40
50 10
198573618294842 159819849819205 698849290010992 000000000000000 139581938009384 158919111891911
182731827381787 135788359198718 187587819218927 185783759199192 857819038188122 897387187472737
159938981818247 128974182773177 135885818282838
57 1984 577 3001 2003 2984 5988 190003 9000 102930 5938 1000000 1000000 5892 38
02111111 10711111 11071111 11104111 41110111 11111031 11111107 11111210
8 28 28 28 28 28 28 28
```

Output Example

```
218
205
48
260445
92
```