19^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

21 October 2017

Problem A — Yellow Balloon Alphabetics

Problem Description

You have been asked to perform a simple lexical analysis of works of literature. As part of this analysis you need to determine how many words there are in a given piece of text. In addition you need to count how often each of the alphabetic characters (`a`–`z`) occur within the given piece of text. The text is known to not only contain the lowercase characters `a`–`z`, but also the uppercase characters `A`–`Z` as well as various punctuation characters `. , ? ' : ; -`. Some of the text isn't well formatted and there may be multiple spaces between some of the words.

A word is defined as one or more non-whitespace characters, that contains at least one alphabetic character (`A`–`Z` and/or `a`–`z`).

No distinction should be made between an upper case and lower case character, for example, both `A` and `a` count as the same character.

Input

The input consists of an arbitrary number of records, but not more than 20.

Each record consists of a single line (terminated by a newline character - `\n`), representing a piece of text containing one or more words (not more than 10 000 characters and 1 000 words), which only consists of the characters listed above. There will not be any leading or trailing spaces on the lines.

The end of input is indicated by a line containing only `-1`.

Output

For each record, output a line with the first value indicating the number of words in the piece of text, followed by the number of times each of the 26 alphabet characters appeared within the text, $c_a \ldots c_z$. Uppercase and lowercase characters should be counted together, all other characters must be ignored.

Sample input

```
The apple falls under the tree.
Careful, it can't be that easy
So ... you thought it was  obvious?
-1
```

Sample output

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

Time limit

1 second

19^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

21 October 2017

Problem B — Pink Balloon
Ancient Collider

Problem Description

While excavating old ruins in the Amazon jungle, you've come across an underground tunnel that forms a large circle, filled with strange machinery. You have finally found the site of the fabled Mayan particle accelerator! Before starting to flip any switches to see if it can be turned on (and hopefully avoiding any of the traps which will kill you with high-powered lasers), it's important to make sure you know exactly where in the tunnel you are.

The tunnel has mystic symbols carved into the walls at regular intervals. Your assistants have made a map that shows all the symbols encountered when walking clockwise around the tunnel, starting from some arbitrary point. Unfortunately, the same mystic symbol may occur in more than one place in the tunnel, so it might not be enough to just look at the nearest symbol. You may need to walk around the tunnel looking at symbols until you have enough information to be sure you know where you are on the map. It might even be impossible (for example, if all the symbols are the same).

You want to know the shortest possible time in which you can be sure you know where you are, if it is possible. It takes you one minute to walk from one symbol to an adjacent one. Ignore the time needed to walk from where you start to the nearest symbol. Note that the symbols are positioned so that they can only be read when standing directly in front of them.

As an example, suppose the map reads `AAB` (where each English letter represents one mystic symbol). In this case, one minute suffices: if you start at a `B`, you already know where you are, while if you start at an `A`, then you can move to the next symbol in either direction to have enough information to know where you started.

Input

Input consists of an arbitrary number of maps, but no more than 25. Each map consists of a line containing upper-case English letters (A to Z), which represent the symbols on the tunnel in the order they appear walking clockwise around the tunnel. Each map contains between 3 and 1000 symbols, inclusive.

The end of input is indicated by a line containing only the value -1.

Output

For each input map, output a line containing the minimum time (in minutes) in which you can guarantee that it will be possible to know your location, regardless of where you started. If it is impossible to ever know your location, output -1 for that map instead.

Sample input

```
AAB
ABCABC
ABCABCD
CABBABBAAA
-1
```

Sample output

```
1
-1
2
3
```

Time limit

45 seconds

19^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

21 October 2017

Problem C — Green Balloon Equations

Problem Description

A student has been given the numbers in an equation, but the operations have all been erased. They must work out what the operations were to make the equation correct. Due to the sheer number of these equations that they have to work through, you have been approached to help.

Fortunately there are only four operations possible, addition (+), subtraction (−), multiplication (*) and division (/). The usual precedence of operations applies, i.e., multiplication and division are computed before addition and subtraction.

The equations are of the form

$$A = B_1 \circ B_2 \circ B_3 \circ \cdots \circ B_n,$$

where A and B_i are integers, and $\circ$ represents the positions where operations need to be inserted.

Division may only be used if integer division is possible without remainders and without dividing by zero (for example, 5/3 is not allowed, but 6/3 is).

You are guaranteed that exactly one solution exists for each equation.

Input

The input consists of an arbitrary number of equations, but not more than 100. Each record consists of a single line, with a list of numbers on it. The first number n ($2 \leq n \leq 10$) is the count of numbers on the right of the equals sign. The second number on the line represents A, and is followed by n numbers representing B_1, B_2, ..., B_n.

The end of input is signified by -1 on its own line.

All input values are non-negative integers, and it is guaranteed that the product of the non-zero B_i's is at most 10^{18}. The `long` type in Java and `long long` type in C/C++ are sufficient to store such values.

Output

For each input record, output the equation that results, in the form shown in the sample output. There must be a single space between each number and any adjacent operators.

Sample input

```
4 16 2 2 2 2
3 9 2 3 3
3 21 1 4 5
-1
```

Sample output

```
16 = 2 * 2 * 2 * 2
9 = 2 * 3 + 3
21 = 1 + 4 * 5
```

Time limit

2 seconds

19^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

21 October 2017

Problem D — Blue Balloon
Knots of Fun

Problem Description

A number of people are sitting on planks around a central pole out over the ocean. They are numbered from 1 to n, where the person to right of person i is $i + 1$ (for $i < n$) and the person to the right of person n is person 1. Each plank is held up by a number of knots. Each person's objective is to untie the knots holding up their right-hand neighbour's plank, so that the plank drops and the neighbour falls into the ocean. Some planks may have more knots than others, and some people may be quicker at untying knots than others. In the end there may be some people still sitting on their planks.

Each person can only reach the knots on their right-hand neighbour's plank. Once they have untied all these knots, they just sit back and wait, hoping their own knots are not untied — they cannot stop their own knots from being untied and they cannot untie anyone else's knots.

Input

The input consists of an arbitrary number of records, but not more than 20.

The first line of each record indicates the number of people sitting on planks, n ($2 \leq n \leq 100\ 000$). The second line of each record has n values $p_1, p_2, \ldots, p_n$, where plank i is held up by p_i knots ($1 \leq p_i \leq 1\ 000$). The third line of each record has n values $t_1, t_2, \ldots, t_n$, where person i requires t_i minutes to untie a single knot ($1 \leq t_i \leq 100$).

The end of input is indicated by a line containing only the value -1.

Output

For each record, output a line listing the people (in ascending order) who are still sitting on their planks after everyone has either fallen into the ocean or finished untying the knots on the plank to their right. If no one is left sitting on a plank, output a line containing 0 for that record.

Sample input

```
6
6 5 4 3 2 1
1 1 1 1 1 1
4
2 7 2 7
1 4 1 4
3
2 2 2
3 3 3
-1
```

Sample output

```
1
1 3
0
```

Time limit

5 seconds

19^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

21 October 2017

Problem E — Purple Balloon Long Division

Problem Description

The Independent College for Primary Children down the road from you has decided to launch a programme to teach its grade 3 learners basic mathematical skills. They have found that the learners struggle with long division problems and approached you to write a program that given the numerator and denominator, will produce a step-wise solution.

The process of long division works as follows (using numerator 124, and denominator 3 as example):

1. First the denominator is written down, followed by a `|` and then the numerator	`3|124`
2. Next, a `+` is aligned above the `|`, followed by a `-` over every digit of the numerator	`  +---` `3|124`

3. At this point you take the most significant (further left) digit of the numerator, and divide that by the denominator, and write the answer directly above the digit that was divided (1 divided by 3 is 0)	`  0`
	`  +---`
	`3|124`

4. Next up you take the result (0), and multiply it with the denominator (call this product m), and write that directly beneath the digit that was divided, and draw a line beneath that using `-`	`  0`
	`  +---`
	`3|124`
	`  0`
	`  -`

5. You then take the original digit that was divided, and from that subtract m. Note that if the result here is zero nothing should be written unless this is the last digit and there are no further digits to carry down	`  0`
	`  +---`
	`3|124`
	`  0`
	`  -`
	`  1`

6. Now you simply carry the next most significant digit down	<pre> 0
	 +---
	3|124
	  0
	  -
	  12</pre>

7. At this point you take what you've got at the bottom, and divide that by the denominator (which is guaranteed to be less than 10), and write this next to the previous answer digit. In the example 12 divided by 3 is 4, which goes next to the initial 0.	<pre> 04
	 +---
	3|124
	  0
	  -
	  12</pre>

8. At this point you simply repeat from step 4, using only the just added digit for the multiplication, you repeat this process until there are no further digits to carry down	<pre> 041
	 +---
	3|124
	  0
	  -
	  12
	  12
	  --
	  4
	  3
	  -
	  1</pre>

9. In the above case the final post-subtraction number is non-zero and the division thus has a remainder, which needs to be shown as part of the answer, by adding a `r1` (remainder 1, this 1 needs to be whatever the remainder is), and extending the line directly beneath the answer to also be under the remainder	<pre> 041r1
	 +-----
	3|124
	  0
	  -
	  12
	  12
	  --
	  4
	  3
	  -
	  1</pre>

A few notes:

- The horizontal line, for subtraction, spans exactly under the value of m.
- When subtracting the only time a zero-digit (0) should be printed is if it's the last subtraction.
- Leading 0s in the answer are not discarded.

Input

Input consists of an arbitrary number of records, but no more than 20.

Each record consist of two numbers, the numerator (n) and the denominator (d). As these are young learners, restrictions of have been placed on the numerator and denominator, $2 \leq n \leq 144$ and $1 \leq d \leq 9$.

The end of input is indicated by a line containing only the value -1.

Output

You need to show the result of the long division in the format that the teacher taught the learners. Each long division problem will be represented by a number of lines of working out. Output `-*-` on a new line after each result.

WARNING: White space in this problem is important. Without this alignment won't be correct, and your solution will be marked as incorrect. This includes trailing white spaces, which must be avoided.

Sample input

```
10 2
124 3
40 4
144 9
-1
```

Sample output

```
  05
 +--
2|10
  0
  -
  10
  10
  --
   0
-*-
  041r1
 +-----
3|124
  0
  -
  12
  12
  --
    4
    3
    -
    1
-*-
  10
 +--
4|40
  4
  -
   0
   0
   -
   0
-*-
  016
 +---
9|144
  0
  -
  14
   9
   -
   54
   54
   --
    0
-*-
```

Time limit

5 seconds

19^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

21 October 2017

Problem F — Red Balloon Pizza

Problem Description

Bob's pizza restaurant offers a number of pizzas on its menu, but also allows customers to modify their pizzas. Each topping has a value. A customer starts by selecting a pizza from the menu, and then makes a sequence of zero or more of the following modifications:

- A topping may be added to the pizza. The price of the pizza increases by the value of the topping.
- A topping may be exchanged for another topping of the same or lower value. The price of the pizza is unchanged.
- A topping may be removed. The price of the pizza is unchanged.

Toppings may appear more than once on a pizza to obtain a larger quantity of that topping.

There are many ways in which one could build a particular pizza, and Bob's miserly customers want to find the cheapest. Given the menu, the values of toppings, and a desired pizza, determine the minimum possible price for that pizza.

Input

Input consists of an arbitrary number of records, but no more than 10. The first line of each record contains two integers n ($1 \leq n \leq 100$), the number of possible toppings, and m ($1 \leq m \leq 100$), the number of predefined pizzas on the menu.

The next n lines describe the possible toppings. Each line contains an integer v ($1 \leq v \leq 1\ 000$), the value of the topping in cents, and a string of up to 20 lower-case English letters, the name of the topping. No two toppings in a record have the same name.

The next m lines describe the menu, one pizza per line. The line starts with two integers p ($1 \leq p \leq 10\ 000$) and t ($0 \leq t \leq 100$), which are the price of the pizza in cents and the number of toppings. This is followed by the names of the t toppings.

The final line of the record contains an integer u ($0 \leq u \leq 100$), the number of toppings on the customer's desired pizza, and u names of toppings.

The toppings on every pizza in the record are guaranteed to appear amongst the n possible toppings.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output a line containing the minimum possible cost in cents for the customer's desired pizza.

Sample input

```
4 2
40 garlic
40 peppers
80 pepperoni
80 mince
600 2 garlic peppers
620 1 pepperoni
2 mince garlic
3 1
45 cheese
70 sausage
60 spam
500 4 spam sausage cheese spam
5 sausage spam spam sausage spam
-1
```

Sample output

```
660
630
```

Time limit

2 seconds

19^{th} South African Regional ACM Collegiate Programming Contest

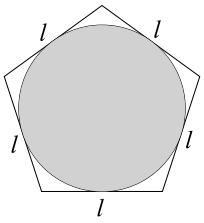
Sponsored by IBM

21 October 2017

Problem G — White Balloon
Polygons to Circles

Problem Description

As part of an engineering project you're busy with, you need some perfectly round metal disks. Unfortunately the only pieces of metal available to you are regular polygons (i.e. all sides are equal in length and all angles are equal). You need to determine the largest circle you can cut out of each regular polygon, assuming a zero-width cutter.



Input

The input consists of an arbitrary number of records, but no more than 30.

Each line of input consists of two integers, n and l where n is the number of sides the polygon has, $(3 \leq n \leq 50)$, and l is the length of the sides of the polygon, $(1 \leq l \leq 1\ 000)$.

The end of input is indicated by a line containing only the value -1.

Output

For each record, output the diameter of the largest circle that will fit within the regular polygon, output as a decimal number with 3 decimal places.

Sample input

```
3 5
8 15
-1
```

Sample output

```
2.887
36.213
```

Time limit

1 second

19^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

21 October 2017

Problem H — Orange Balloon Road Trip

Problem Description

Every year for your holiday, you take a road trip through the Republic of Treeland. Treeland consists of N towns, with roads connecting them. The people of Treeland are minimalists: for every pair of towns in Treeland, there is exactly one route between them. Also, each town has exactly one airport, and either one museum or one art gallery. Each year you fly into some town, drive to a different town along the unique route, and fly home again. Along the way, you visit every museum and art gallery (including those in the start and end towns).

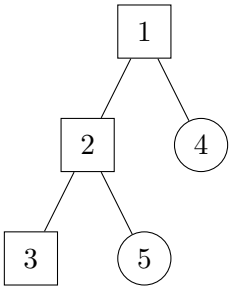


Figure 1: *A possible road map in Treeland, corresponding to the first sample input. Squares represent towns with museums and circles represent towns with art galleries.*

You get bored with too much of the same thing, so you don't want to visit more than two museums in a row or more than two art galleries in a row. For example, in Fig. 1 you could make the trip 4–1–2–5, but not 4–1–2–3 because 1–2–3 would be three museums in a row.

You also prefer to drive a different route every year, so you would like to know how many valid routes there are. Driving from A to B and from B to A is considered the same route.

Input

Input consists of an arbitrary number of records, but no more than 20.

The first line of each record contains n ($2 \leq n \leq 100\ 000$), the number of towns in Treeland. The towns are numbered from 1 to n. The next line contains n integers, where the ith integer is 0 if town i has a museum and 1 if it has an art gallery. This is followed by a line containing $n-1$ integers $p_2, p_3, \ldots, p_n$, with $1 \leq p_i < i$. For each i ($1 < i \leq n$) there is a road between towns i and p_i.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output a line containing the number of routes that satisfy the conditions.

Sample input

```
5
0 0 0 1 1
1 2 1 2
6
1 1 0 1 0 1
1 1 1 1 1
-1
```

Sample output

```
8
12
```

Time limit

10 seconds