International Collegiate Programming Contest 2022

Latin American Regional Contests

March 18, 2023

Warmup Session

This problem set contains 3 problems; pages are numbered from 1 to 6.

This problem set is used in simultaneous contests with the following participating countries:

Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, El Salvador, México, Perú, Puerto Rico, República Dominicana, Uruguay and Venezuela

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp`, *codename*`.java`, *codename*`.kt`, *codename*`.py3`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. The English alphabet must be used. There should be no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.
6. To output real numbers, round them to the closest rational with the required number of digits after the decimal point. Test case is such that there are no ties when rounding as specified.

Problem A – Almost Origami

You have a rectangular sheet of paper of height 1 and you want to locate any point at height H measured from the bottom border of the sheet. Since you do not know Haga's theorems, you plan to repeat the following step. Assume you already located a point P_L at height L on the left border of the sheet, and a point P_R at height R on the right border of the sheet. Then you draw a line from the lower left corner of the sheet to P_R, and another line from the lower right corner of the sheet to P_L. If the crossing point is at height H, then you are done. Otherwise you draw a horizontal line that passes through the crossing point and go for another step.

As an example, consider the case $H = 1/3$. During the first step, the only possibility is choosing the upper corners of the sheet (that is, $L = R = 1$). So you draw the two diagonals of the sheet, and the crossing point is at height $1/2$. Since $H \neq 1/2$, you draw a horizontal line that passes through the crossing point. This line provides two new points with known height $1/2$ on the borders of the sheet, one on the left border and the other one on the right border. For the second step you can choose between using the original known points at height 1, or the points you have just located at height $1/2$. That is, you can choose either $L = 1$ or $L = 1/2$ and of course $R = 1$ or $R = 1/2$. It is easy to see that if you choose $L = R = 1/2$, then the crossing point would be at height $1/4$. However, if you choose $L = 1/2$ and $R = 1$, then the crossing point would be at the desired height $H = 1/3$. By symmetry, the same occurs if you choose $L = 1$ and $R = 1/2$.

Given a rational height H, you must determine a shortest sequence of heights on the borders of the sheet that allows locating a point at height H.

As the above example shows, only a point at height $1/2$ can be located in a single step, and so a possible shortest sequence for $H = 1/3$ is $S = (1, 1, 1/2, 1)$. The first two heights must be chosen during the first step, and the remaining two heights must be chosen during the second step.

Input

The input consists of a single line that contains two integers M and N ($1 \leq M < N \leq 100$) such that $H = M/N$ is an irreducible fraction.

Output

Output a single line with the character "*" (asterisk) if a point at height H cannot be located by means of the described procedure. Otherwise, output a shortest sequence of heights $S_1, S_2, \ldots, S_K$ that allows locating a point at height H, if they are chosen in the order they appear in the sequence. Height S_i must be written in the i-th line using two integers A_i and B_i such that $S_i = A_i/B_i$ is an irreducible fraction ($i = 1, 2, \ldots, K$). It is guaranteed that when a point at height H can be located, it can be optimally located choosing only rational heights.

Sample input 1	**Sample output 1**

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

Sample input 2	**Sample output 2**

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

Sample input 3	**Sample output 3**
`3 4`	`*`

Sample input 4	**Sample output 4**

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

Problem B – Invested Money

Nowadays your programming skills are amazing, and you regularly receive lots of money for your work. Unfortunately, your financial skills did not evolve the same way. So each time you earn some money, you simply invest it in a bank in a 30 days time deposit with an automatic renewal clause. This means that 30 days after you invest the money, it is invested for 30 additional days, over and over again, until you inform the bank that you want to stop the renewal and get your money back. Time deposits cannot be created nor renewed during weekends; if a 30 days period ends on a weekend, the renewal occurs on the immediately following Monday.

Since the bank holds almost all your money, you must wait until the closest renewal each time you want to buy anything but daily food. Today you decided to buy a new smartphone to replace your six-month-old device. Given the dates when you created each time deposit, you must determine the minimum number of days that you must wait to get some money from the bank.

As an example, suppose that today is Saturday and that you created five time deposits: a time deposit last Monday, another time deposit last Tuesday, yet another time deposit last Wednesday, and two time deposits yesterday. The first time deposit (Monday) would be renewed on a Wednesday after 25 days from today. The second time deposit (Tuesday) would be renewed on a Thursday after 26 days from today. The third time deposit (Wednesday) would be renewed on a Friday after 27 days from today. Finally, the last two time deposits (Friday) would be renewed on a Monday after 30 days from today, because the renewal on a Sunday is not allowed. Thus, in this case, you must wait 25 days to get some money from the bank.

Input

The first line contains a string T and an integer N ($1 \leq N \leq 10^5$), indicating respectively today's day of the week and the number of time deposits. The string is either "`Mon`", "`Tue`", "`Wed`", "`Thu`", "`Fri`", "`Sat`", or "`Sun`", representing respectively that today is Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday. The second line contains N integers $D_1, D_2, \ldots, D_N$ ($0 \leq D_i \leq 10^9$ for $i = 1, 2, \ldots, N$), indicating the number of days elapsed since each time deposit was created. It is guaranteed that the time deposits were not created during weekends.

Output

Output a single line with an integer indicating the minimum number of days that you must wait to get some money from the bank.

Sample input 1	**Sample output 1**

`Sat 5`
`5 4 3 1 1`
	`25`

Sample input 2	**Sample output 2**

`Sat 5`
`3 1 4 1 5`
	`25`

Sample input 3	**Sample output 3**

`Thu 1`
`0`
	`32`

Sample input 4	**Sample output 4**

`Thu 1`
`30`
	`0`

Sample input 5	**Sample output 5**

`Fri 1`
`31`
	`31`

This page would be intentionally left blank if we would not wish to inform about that.

Problem C – Non-Integer Donuts

Neil is a very important lawyer with a very important bank account. Since Neil is such a successful lawyer with many clients, he deposits money to his account every single morning.

After going to the bank and depositing money, Neil goes to work. And there lies Neil's great weakness: a donut shop. You see, Neil is a recovering donut addict, and although he hasn't eaten a donut in years, he can't help but wonder how many \$1.00 donuts he could buy with the money in his account if he were to relapse.

Having \$5.00 in his account means 5 donuts Neil could have, but what about \$4.50? Well, that is more than 4 donuts for sure, but definitely less than 5. How would one even buy a non-integer amount of donuts? That concept confuses Neil, so every time his account balance is not an integer, he stops to ponder the nature of non-integer donuts and ends up being late to work.

Now Neil has been late too many times and is starting to worry he will lose his job. He wants to know how many times he will be late to work during the next N days, given his initial account balance and the amount of money he will deposit each day. Please answer this for him, or else Neil will start pondering again.

Input

The first line contains an integer N ($1 \leq N \leq 1000$), the number of days Neil is interested in. Each of the next $N + 1$ lines contains a string representing an amount of money. The first string is Neil's account initial balance, while the following N strings are the amounts Neil will deposit to his account in the different days. Each string has the form $\$X.Y$ where X is a substring of length 1 or 2 indicating the whole money in the amount $\$X.Y$, while Y is a substring of length exactly 2 denoting the cents in the amount $\$X.Y$. Both X and Y are made of digits, at least one of them contains a non-zero digit, and X does not have leading zeros.

Output

Output a single line with an integer indicating how many times Neil will be late to work during the following N days.

Sample input 1	**Sample output 1**

`1`
`$1.57`
`$3.14`
	`1`

Sample input 2	**Sample output 2**

`4`
`$1.00`
`$0.01`
`$0.99`
`$10.00`
`$98.76`
	`2`

This page would be intentionally left blank if we would not wish to inform about that.