Weekend Contest

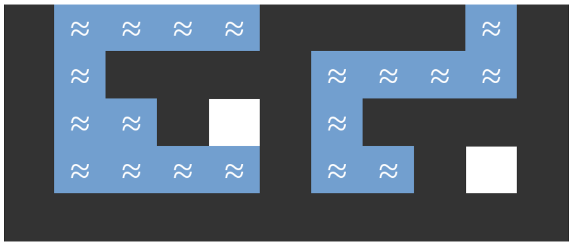
A. Aquarium Maze

3 seconds, 256 megabytes

Again, you are sitting in the lecture hall 037 and enjoying a thoroughly exciting lecture; but you can't help it, your mind starts to drift into the depths of your thoughts. At first you wonder why the professor picked that font in the slides, then you remember that you forgot to do the maths homework for yesterday. After some more deep insights, most of which you have already left behind you in a forgotten realm, you catch yourself in the blue with the remarkably appealing question: "What if the wall behind the professor was made of an aquarium?!"

It is too late to listen now, you note to yourself, this is now a question of utmost priority. You start the inception process of such an undertaking, making various improvements on the plan. You realize that a plain aquarium is boring, what if it had various walls inside of it? And then you grasp that if you were to manufacture it and then starting pouring in water from the top, that various parts of the aquarium might never be filled with water, because air-pockets would form! You wonder, what is the maximum amount of water that such an aquarium can contain?

You are given a grid representing a 2D aquarium, each character represents 1 litre of volume. You may add water to it using any of the air parts in the first row of the aquarium. The water added in such a way, will then either flow left, right, or down in the aquarium (it will not move up into air-pockets). What is the maximum number of litres you can add to the aquarium by filling it with any of the air openings in the first row?


Solution for sample 2

Input

The first line contains 2 numbers, r the number of rows ($3 <= r <= 100$), and c the number of columns ($3 <= c <= 200$) in the aquarium.

Then r rows follow, each with c characters consisting of air `.` or glass #. It is guaranteed that the left and right columns and the bottom row consist of only glass # characters (i.e. no water can exit the aquarium once added).

Output

Print a single integer, the maximum number of litres that you can fill into the aquarium if you fill it from all of the openings in the first row of the aquarium.

input
```
7 18
#................#
#................#
#...##########...#
#...#........#...#
#...#........#...#
#................#
##################
```
output
```
66
```

input
```
5 11
#....####.#
#.####....#
#..#.#.####
#....#..#.#
###########
```
output
```
19
```

input
```
3 3
###
#.#
###
```
output
```
0
```

B. Bicycle Lock

3 seconds, 256 megabytes

Based on a real story.

You just arrived at the Konrad-Zuse-Haus, it is already 17:13! Your lecture starts in 2 minutes and you still have to lock your bike. Unfortunately it is winter, so you are wearing gloves, which means that inputting the correct code into your combination bicycle lock will be hard! You realize that because of your gloves you can only turn *exactly* **2** dials at once. You bite the bullet this time, sacrificing your bare hands to the bitter cold. But as you step in the 037 lecture hall you keep wondering, was it possible to unlock the lock without undoing your gloves? And if yes, how many dial turns would you have needed?

- A dial always contains 10 numbers, 0 to 9. It rotates in such a way that 0 and 9 are adjacent.
- A dial turn is considered 1 rotation of 2 adjacent dials by 1 step. For example, to go from `3 9 1` to `5 1 1` is 2 dial turns.

Input

You are given 3 lines:

- The first line contains an integer n ($2 <= n <= 100$), specifying the number of dials on your bicycle lock.
- The second line contains n integers c_i ($0 <= c_i <= 9$) the current position of the n dials.
- The third line contains n integers t_i ($0 <= t_i <= 9$) the target position of the n dials.

Output

Print "impossible" if it is not possible to turn the dials without undoing the gloves. Otherwise print the minimum number of turns to reach the target position.

input

```
4
1 4 6 1
1 6 1 4
```

output

```
5
```

input

```
2
9 0
5 6
```

output

```
4
```

input

```
3
9 0 1
5 6 2
```

output

```
impossible
```

C. Compilers & Brackets

1 second, 256 megabytes

You just completed the lecture on how to create compilers by professor Wolf, and now that you are an expert at it you want to test your skills. You decide to create your own programming language. First steps first, however, so you decide the most important thing is brackets, and lots of them. You want to write a parser which tells you whether a string of open and closed brackets is valid.

A string of brackets is defined as valid if:

- Each open bracket is closed by a closed bracket.
- A closed bracket can only close a single open bracket.
- No closed bracket exists that does not close an open bracket.

- "{" - is an open bracket.
- "}" - is a closed bracket.

Input

A single line containing a string s $(1 <= ||s|| <= 1000)$ entirely consisting of the characters "{" and "}".

Output

Print "valid" if the given string is a valid sequence of brackets, otherwise print "invalid".

input

```
{{}}{}
```

output

```
valid
```

input

```
{}{}{{}}}
```

output

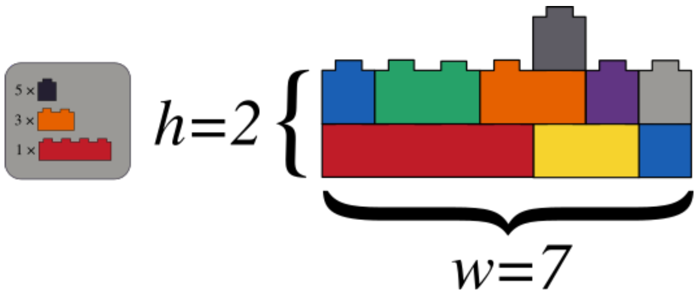
```
invalid
```

D. Dam Construction

0.5 seconds, 256 megabytes

You are the manager for a dam construction company called *Damn Dams Inc.*, and now you have a new contract. Luckily for you, the company is based in Rostock, so the dams are rather small. In fact, they are built from lego bricks! You are to oversee the construction of a new dam, you are not sure though whether you have enough materials (lego bricks) for it.

Your company only uses three types of bricks to build dams: 1x1, 2x1 and 4x1 bricks (see picture). The contract requires the dam to be of exact width w. In order to estimate if you can build the required dam, you need to evaluate your available resources first. What is the maximum height you can build the dam to using the available resources, such that there are no holes? Only count complete layers for this. It does not matter how bricks in two adjacent layers overlap, because you will be using *Damn Dams Inc.* patented sealant for structural integrity.



Example solution for sample 1

The bricks are all of the same height and cannot be rotated, due to the pins on the bricks. No brick may extend the given width w.

Input

The first line contains a single integer w $(1 <= w <= 10^9)$, the exact width which each layer of the dam needs to be. The second line contains three integers b_1, b_2 and b_4 $(0 <= b_1, b_2, b_4 <= 10^9)$, the number of bricks of width 1, 2 and 4 respectively.

Output

A single integer: the maximum height h which is possible to build to with complete layers with the given bricks.

input

```
7
5 3 1
```

output

```
2
```

input

```
5
3 0 500
```

output

```
3
```

input

```
86
186496674 506784125 358221693
```

output

```
30615717
```

E. Extravagant Journey

5 seconds, 256 megabytes

You are travelling home in a couple of hours (as you have read in Problem *G. Going Home*) and are packing your belongings in your backpack. You have had enough time before the first train to ask your mother if she has any wishes for her birthday, and she has told you numerous items she wants. Unfortunately, your backpack only has enough space for m millilitres of volume, so you may not be able to bring all of the items she has asked for.

In order to solve this dilemma you have created a list of each item she has asked for. You have figured out the volume of each item in millilitres and you have assigned a score to each item: The number of hours that item will make her happy for. Now you wonder if there is a way to maximize the happiness of the items that you bring with you? I.e. what is the maximum sum of scores that you can bring with you?

Note that your mother will of course be happy since you are visiting at all, but you deem that this happiness is a constant and therefore is not needed to account for in the final score.

Input

The first line contains 2 integers n $(1 <= n <= 1000)$, the number of items and v $(1 <= v <= 1000)$ the volume of your backpack in millilitres.

The second line contains n integers v_i $(1 <= v_i <= 1000)$, the volume of the i-ith item in millilitres.

The third line contains n integers s_i $(1 <= s_i <= 100)$, the score of the i-ith item (how many hours it makes your mother happy).

Output

Print a single number, the maximum number of hours that you can make your mother happy.

input

```
4 21
10 17 13 5
2 18 3 16
```

output

```
19
```

input

```
5 1000
995 4 4 2 2
1 30 31 20 19
```

output

```
100
```

F. Fascinating Books

1 second, 256 megabytes

You are again on the hunt for the best programming language there is, this time, it's Visual Basic. And since you are incredibly up to date with the times you are looking in the university library for a good book on it. However, after looking at hundreds of books, every title you read starts to lose its meaning, and you see just the letters. You see that some letters appear more often than others, and you start to wonder, do all letters of the English alphabet appear at least once on this shelf?

Input

The first line contains an integer n $(1 <= n <= 50)$, the number of book titles on the shelf. Then n lines follow, each containing a title of a book on that shelf. Each book title will have between 1 and 100 characters (1 and 100 included). Each character will be one of `a-z`, `A-Z`, `:'` (colon or a single qoute) or a space.

Output

Print `yes` if the books contain each character of the English alphabet at least once or `no` if they don't.

input

```
7
Visual Basic
Introduction to Algorithms
Computational Complexity
Reviewing Java
Exploring Requirements: Quality Before Design
Structured Computer Organization
Hacker's Delight
```

output

```
yes
```

input

```
3
Introduction to Algorithms
The C Programming Language
Structured Computer Organization
```

output

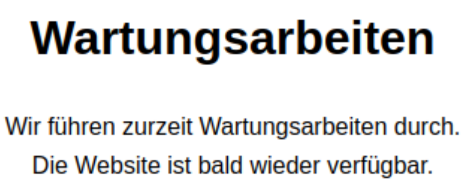
```
no
```

G. Going Home

3 seconds, 256 megabytes

It just turned 00:00; you look at your phone: a new calendar notification just popped up. It reads *Mom's birthday @ 09.01.2023*, you are in shock, how could you have forgotten! You promised to be at home, but now you only have 24 hours to plan and get there from Rostock! So you open the site for the Deutsche Bahn, but you frown, it's in maintenance. Luckily you are a cunning programmer*, so you check it's API, and you see it's still up! However, the API only returns single train connections without transfers, not full routes. You decide to take matters into your own hands. Given a list of connections find the earliest time you can arrive at your home destination (it is guaranteed that you can arrive 23:59 or earlier, i.e. all connections depart and arrive on the same day).

* Assume that you are not cunning enough to check any of the other sites which may return a complete route!


Wartungsarbeiten

Wir führen zurzeit Wartungsarbeiten durch.
Die Website ist bald wieder verfügbar.

Completely fictitious scenario

Input

The first line contains two integers and a string:

- the number of cities n $(2 <= n <= 200)$,
- the number of directional connections in total m $(1 <= m <= 10000)$ and
- a city name h consisting of lowercase English alphabet letters $(1 <= ||h|| <= 10)$. This city name denotes the target city where your mom lives.

Then m lines follow with the following format:

`from_city to_city mins trains [connection]*trains`. E.g.:

`rostock dresden 50 2 10:21 12:21`.

The above line means that there are 2 trains leaving from `rostock` to `dresden`, at 10:21 and at 12:21. They will arrive at 11:11 and 13:11 respectively.

$from\_city$ and to_city are strings of lowercase english alphabet letters with a length of between 1 and 10 inclusive. $mins$ $(1 <= mins <= 1440)$ is the interval between trains in minutes for this connection. $trains$ $(1 <= trains <= 20)$ is the number of departure times that follow. Then $trains$ number of strings follow, formatted as `HH:MM`.

Output

A string of length 5 formatted as (`HH:MM`), the earliest time of arrival at your home city *h* from the city of `"rostock"`.

input
```
5 5 magdeburg
rostock berlin 129 3 08:21 10:21 12:21
rostock schwerin 53 6 08:25 09:08 10:25 11:08 12:25 13:08
berlin magdeburg 100 6 08:11 09:11 10:11 11:11 12:11 13:11
schwerin wittenberge 59 3 08:00 10:00 12:00
wittenberge magdeburg 94 3 09:10 11:10 13:10
```
output
```
12:44
```

input
```
2 1 dresden
rostock dresden 254 5 02:21 06:21 10:21 14:21 18:21
```
output
```
06:35
```

H. Hidden Words

2 seconds, 256 megabytes

As part of the operating systems course you have designed your own ingenious protocol for transmitting a list of words. You have noticed that transmitting spaces is a waste of data, since everyone obviously can decipher where they should be, so you cut those. And then you notice that some words share letters, so in order to save precious bytes you only send that part of the word once. For example, given the words *lint* and *inter*, you may send *linter*, because both words share *int*.

However, you deem that the first character of a word that comes later in the sequence cannot appear before the first character of a previous word, because otherwise it would not be clear in which order the words are! For example, given the words *bob*, *at* and *bat*, you may not print *bobat*, because *bat* would begin before *at* despite appearing later in the sequence (*bobatbat* would be correct).

Input

The first line contains an integer n $(1 <= n <= 1000)$, the number of given words. The second line contains n words w_i, each separated by a space. Each word w_i will have at least 1 and at most 50 characters. All characters are lowercase english letters.

Output

Print the string of minimal length, which appears when you apply your ingenious protocol.

input
```
6
in inter intercontinental continentally tallying unrelated
```
output
```
intercontinentallyingunrelated
```

input
```
4
at bat attack king
```
output
```
atbattacking
```

I. Intuitive Citations

1 second, 256 megabytes

You have submitted your first paper for review, unfortunately it has been rejected. But the only critique is that you have not followed the citation guidelines. Instead of listing all authors you are required by the journal to just list the first surname alphabetically and add a `"et al."`. You have hundreds of citations, so you decide to automate this process. Write a program which takes all the authors of a work and prints the first surname when ordered alphabetically and adds `"et al."`.

Input

The first line contains an integer n $(1 <= n <= 50)$, the number of authors. Then n lines follow, each with 2 strings, the first name and last name of each author. Both will only consist of letters of the English alphabet, all letters will be lowercase, except for the first letter, which will be uppercase. Both strings will have between 2 and 20 characters (inclusive).

Output

Print the first surname in alphabetical order and add `" et al."`

input
```
4
Donald Knuth
Edsger Dijkstra
Allen Newell
Ken Thompson
```
output
```
Dijkstra et al.
```

input
```
1
Alan Turing
```
output
```
Turing et al.
```

J. Jolly Fishing

1 second, 256 megabytes

As part of your new job in the digitalisation of the government of Rostock you are tasked to implement a responsible fishing policy, "just like the Norwegians do", you were told. You are supposed to set a policy for a year (365 days) for fishers where you maximize the number of fishes caught, but you make sure that at the end of the year there is still at least as many fish as at the start of the year.

On each day you can give the fishers a permit to fish, otherwise they are not allowed to fish, and since this is Germany you are sure they won't. On each day of the 365 days the fishers will fish exactly c‰ (‰=per mille) of fish (i.e. $c/1000$ of fish will be fished on such a day). On days where fishers are not allowed to fish the fish population will increase by a factor of b per mille (i.e. the fish population will increase by a factor of $b/1000$ fish). On days where fishing is allowed fishes will of course not reproduce, because the fish are too scared when they are being fished.

A fisher calculating his profits, by DALL-E 2

Input

You are given one line with 3 integers:

- a ($1 <= a <= 1000$) the number of fish at the beginning of the year, expressed as units of *Mf* (Megafish = millions of fish)
- b ($1 <= b <= 50$) the growth factor of the fish in per mille (‰)
- c ($1 <= c <= 999$) the fishing factor of the fishers in per mille (‰)

Output

Print a single floating point number, the maximum number of fishes possible to fish in *Mf* (Megafish), in such a way that there are **at least as many** fish after 365 days as there were at the beginning of the year.

In this problem your answer does not have to match exactly with the jury answers, it is enough if your floating point number matches at least to 10^{-6} absolute or relative error. Make sure to use **double-precision** instead of single-precision in order to achieve the required precision.

input
```
2 10 20
```
output
```
29.326586530176023
```

input
```
3 50 1
```
output
```
1214335.7896734662
```

input
```
1000 50 999
```
output
```
51578694986.91669
```

input
```
4 1 999
```
output
```
0.0
```

input
```
1 1 1
```
output
```
0.19988677729070248
```

K. Keyboard Robot

5 seconds, 256 megabytes

Everyone knows that the most important thing for a programmer is their typing speed! And you have been on a journey to improve your typing speed for a while, employing the fastest typing method known to man: 2 finger-typing. You have tested various keyboard layouts, such as Qwerty, Colemak, Dvorak and so on. But none of these are good enough for you, so you have decided to design your own. To streamline this process you have also created a robot which is used to test such a keyboard layout. Of course such a robot only has 2 mechanical fingers, which matches your superior typing preference.

Given a 6x6 keyboard layout you have designed and a sequence of characters, calculate the minimum possible number of seconds needed by 2 mechanical fingers in order to type that character sequence. The fingers start in the top left corner of the keyboard. The fingers can move 1 key up, down, left or right in **1** second, they cannot move diagonally. It takes them **0** seconds to type a character (a finger can immediately move again after arriving at a character it needs to press). After a second has passed you can use **either** finger or **both** fingers to type a letter. In case you use both to type a letter at the same time you have enough control of them that you can type them in either order you want.

Both fingers will be initially at the top left key (even if the key is empty). The fingers do not hinder each other and can stay at or move to the same position. The keyboard does not loop around, therefore the maximum time to move between any two buttons on the keyboard is between the top-left and bottom-right buttons: $5 + 5 = 10$ seconds.

Input

The first 6 lines contain your keyboard layout, each of those lines contains exactly 6 characters. Among these 36 characters it is guaranteed that each letter in the English alphabet and the space-bar appear **exactly** once (in its lowercase form). The space-bar is encoded as a underscore _. The 9 remaining dots (`.`) denote empty parts of the keyboard where there is no key. The fingers can move over or stay on empty parts of the keyboard.

The next line contains a string s $(1 <= ||s|| <= 200)$ consisting of lowercase letters in the English alphabet + spaces (which are encoded as underscores (_)).

Output

Print a single integer number, the minimum possible time in seconds needed to type the given character sequence.

input
```
.rsft.
.qedu.
.pacih
.zbmjg
.yolkn
._xwv.
hello_my_great_friend_i_have_not_seen_you_in_a_long_time
```
output
```
86
```

input
```
abcdef
.....g
stuv.h
r_.w.i
qzyx.j
ponmlk
a_b_c_d_e_f_g_h_i_j_k_l_m_n_o_p_q_r_s_t_u_v_w_x_y_z
```
output
```
28
```

input
```
abcdef
.....g
stuv.h
r_.w.i
qzyx.j
ponmlk
abcdefghijklmnopqrstuvwxyz
```
output
```
22
```

Sample 3 explained:

Type letters `a-k` with finger 1, meanwhile, move finger 2 to the key `l`. Immediately after the first finger types the key k, the second finger can type the key l (now 9 seconds have passed, 11 characters have been typed).

Repeat this process with the letters `m-s` and `t`. Now use finger 2 to type the letters `m-s`, move finger 1 meanwhile to the key `t`. As soon finger 2 types the letter `s`, type the letter `t` with finger 1 (now 16 seconds have passed, 19 characters have been typed).

Same process with `u-x` and `y` (now 21 seconds have passed, and 25 characters have been typed).

Use the last second to move a finger from `y` to the key `z`, getting a time of 22 seconds in total.

L. Leaderboard Prediction

1 second, 256 megabytes

You are a crazy good competitive programmer, so you are participating in the ACM ICPC World Finals 2022 in Egypt. You have h hours for the contest in total. You have already read all n problems, and since you are so good you immediately know how many minutes each problem will take you to solve. Print how many problems you will be able to solve in h hours and what the time penalty will be for those solved problems. Note that you will of course make no mistakes, and each submission will be accepted on the first try.

The scoring for the ACM ICPC World Finals works just the same way as the competition you are participating in right now! Teams are scored by two values, firstly, by the number of correctly solved problems (higher is better), then by the sum of submission times in minutes for the correctly solved problems (lower is better). The second value is called the time penalty. For example, a team which has solved 3 problems at minute 14, 51 and 267 will have a time penalty of $14 + 51 + 267 = 332$. Unsolved problems, even if they have incorrect submissions, do not add to the time penalty.

A problem will count as solved if is solved at or before minute $h \cdot 60$. For example, if the contest is 5 hours long, then a problem can be solved at minute 300 since minute 300 is the last second of the contest at 5:00:00 where you can submit (and that is plenty of time for you).

Input

The first line will contain 2 integers n $(1 <= n <= 15)$ and h $(1 <= h <= 10)$. Then n lines follow, each with a single integer m_i $(1 <= m_i <= 300)$ specifying how many minutes you need in order to solve problem i.

Output

Print 2 integers, the maximum number of problems you can solve during the contest, and the minimum penalty for those solved problems.

input
```
4 1
15
27
5
44
```

output

```
3 72
```

input

```
1 1
240
```

output

```
0 0
```

input

```
1 1
1
```

output

```
1 1
```