# Problem A. Buying in Bulk

| | |
|---|---|
| Source file name: | bulk.c, bulk.cpp, bulk.java, bulk.py |
| Input: | `Standard` |
| Output: | `Standard` |

To encourage customers to shop more, some stores charge lower prices if you buy multiples of an item. For example, if you buy one, it may cost you $5 but if you buy two, it will cost you $8 instead of $10.

Let's assume a store provides discounts as follows:

1. No discount if you buy only one.
2. $2 discount for each additional item if you buy more than one.

Given the number of items a customer has purchased and the price for one item, you are to compute the total cost for the customer.

## Input

There is one input line, it provides two integers: the first integer $c$ $(1 \leq c \leq 100)$ is the number of items purchased by the customer, and the second integer $p$ $(3 \leq p \leq 50)$ is the price for one item.

## Output

Print the total cost for the customer on a line by itself with no leading or trailing spaces.

## Example

| Input | Output |
|---|---|
| `1 5` | `5` |
| `3 10` | `26` |

## Problem B. Are We Stopping Again?

| | |
|---|---|
| Source file name: | stopping.c, stopping.cpp, stopping.java, stopping.py |
| Input: | `Standard` |
| Output: | `Standard` |

Going on a road trip has been an adventure for Dr. Orooji and the programming team. Obviously he has to stop to refuel the car, but he also stops whenever the team members want to eat. Dr. O needs to figure out the number of stops before going on the trip so he is mentally prepared.

Find the total number of stops for Dr. O's trip, given:

1. Total miles to be traveled.
2. How often he stops for gas (in miles).
3. How often he stops for food (in miles).

Assume that the car's gas tank is full at the beginning of the trip and the team members are full as well. If the destination happens to be the time to refuel (or eat), do not count it as a stop. Also, if a particular mileage happens to be both refueling time and eating time, count it as one stop and not two stops.

Note that if a particular mileage happens to be refueling time only, the team members won't eat at that stop. Similarly, if a particular mileage happens to be eating time only, the car is not refueled at that stop.

### Input

There is one input line, it provides three integers (each between 1 and 1000, inclusive); these are the three values specified in order above.

### Output

Print the number of stops for the trip on a line by itself.

### Example

| Input | Output |
|---|---|
| `100 30 40` | `5` |
| `10 5 1` | `9` |
| `20 3 4` | `9` |

# Problem C. Spell Checker

| | |
|---|---|
| Source file name: | spell.c, spell.cpp, spell.java, spell.py |
| Input: | `Standard` |
| Output: | `Standard` |

This question is about automatic spelling correction. Studies have shown that the majority of typing errors are caused by (let's assume the dictionary contains the word "these"):

1. Omitting one letter, e.g., the input word is "thse".
2. Adding an extra letter, e.g., the input word is "thesce".
3. Mistyping one letter, e.g., the input word is "thise".
4. Transposing two adjacent letters, e.g., the input word is "tehse".

With the aid of a dictionary, word processing systems can often detect and automatically correct these kinds of spelling errors.

You are to write a program that recognizes the four kinds of errors described above.

## Input

The first input line contains an integer $d$ ($1 \leq d \leq 100$) indicating the number of words in the dictionary. Each of the following $d$ input lines contains a dictionary word. Assume these words start in column 1, contain at least 1 and at most 15 lowercase letters, and contain no other characters.

Following the dictionary words (i.e., the next input line), there is an integer $n$ ($n \geq 1$) indicating the number of words to be spell checked. Each of the following $n$ input lines contains a word. These words also start in column 1, contain at least 1 and at most 15 lowercase letters, and contain no other characters.

## Output

Print each input word to be spell checked. Then, if the word is in the dictionary, print `CORRECT`. If the word is not in the dictionary, then find each word in the dictionary (in the order provided in the dictionary) for which the given input word might be a misspelling, and print the appropriate message from the following list:

`ONE LETTER OMITTED FROM` *word*

`ONE LETTER ADDED TO` *word*

`ONE LETTER DIFFERENT FROM` *word*

`TWO LETTERS TRANSPOSED IN` *word*

where *word* is a dictionary word. If the input word is not `CORRECT` and none of the above messages apply, then print `UNKNOWN`.

Note that two or more of the above messages might be applicable to an input word, and that one message might apply for more than one dictionary word. Note, however, that for a given input word and given dictionary word, at most one of the above messages apply. For each input word, you are to process the dictionary words in the order provided in the input and print all messages that are valid.

Leave a blank line after the output for each input word. Follow the format illustrated in Example Output.

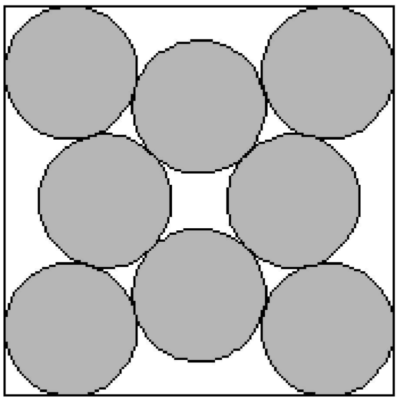
## Example

| Input | Output |
|---|---|
| 7 | ali |
| ali | CORRECT |
| thru |  |
| tu | thu |
| funfunfun | ONE LETTER OMITTED FROM thru |
| the | ONE LETTER ADDED TO tu |
| tuh | ONE LETTER DIFFERENT FROM the |
| th | TWO LETTERS TRANSPOSED IN tuh |
| 3 | ONE LETTER ADDED TO th |
| ali |  |
| thu | orooji |
| orooji | UNKNOWN |

## Problem D. Circles Inside a Square

| | |
|---|---|
| Source file name: | circle.c, circle.cpp, circle.java, circle.py |
| Input: | `Standard` |
| Output: | `Standard` |

You have 8 circles of equal size and you want to pack them inside a square. You want to minimize the size of the square. The following figure illustrates the minimum way of packing 8 circles inside a square:



Given the radius, $r$, find the area of the minimum square into which 8 circles of that radius can be packed.

### Input

There is one input line, it consists of a positive real number (between 0.001 and 1000, inclusive) denoting the radius, $r$.

### Output

Print the area of the minimum square where 8 circles of radius r can be packed. Print 5 digits after the decimal. Your output is considered correct if it is within $\pm 0.00001$ of the judge's output.

### Example

| Input | Output |
|---|---|
| `0.1` | `0.34383` |
| `0.2` | `1.37532` |

## Problem E. Anya's Favorite CD

| | |
|---|---|
| Source file name: | divide.c, divide.cpp, divide.java, divide.py |
| Input: | `Standard` |
| Output: | `Standard` |

A few years ago, Arup started submitting problems to the UCF Local Contest regarding her daughter Anya's CD requests. Unfortunately, his latest question wasn't good enough to make the cut for the 2019 UCF Local Contest. Luckily, a couple coaches, nostalgic for solving Anya's CD problems, suggested that Arup's latest question about Anya's CD requests be added to the practice local contest. Arup is so ecstatic that his question is being used in this contest that he will personally present the first solver with a copy of Ed Sheeran's CD "Divide".

To this day, Arup drives Anya to school daily and Anya makes CD requests in the car. When Anya was four (in 2017), she would request a sequence of track numbers for Arup to play and Arup had to determine the fewest number of button presses to satisfy the requests. If you didn't compete in 2017, here is an explanation of how the CD player in Arup's car works:

Arup can only change tracks by pressing a forward button or a backward button. If track number $k$ has completed, when Arup presses the backward button, track number k will play again. Alternatively, if he presses the forward button when track $k$ has completed, then track $k + 2$ will play. The only exception to the latter is that if track $k + 2$ doesn't exist. In this case, the tracks just wrap back around to the beginning, starting with 1, then 2, etc. Similarly, if Arup presses the backward button twice right after track 1 completes, this will move the CD to track $t$, where $t$ is the number of tracks on the CD. In the absence of a button press, after track $k$ completes, track $k+1$ plays, except for when $k = t$, in which case, track 1 plays next.

Now that Anya is six, her requests have become slightly more flexible. Instead of insisting on particular songs in a sequence, for each song, Anya gives Arup several choices 1 . For example, for the first song, Anya may tell Arup that she wants to hear one of tracks 1, 3, 8 or 9, and for the second song Anya may tell Arup she wants to hear one of tracks 2 or 12.

Even though Arup has some choice in what songs he plays, he still gets caught pressing either the forward or backward button a great deal. Help him minimize the number of times he presses the buttons. In the example above, given that the CD is originally queued to start with track 1, no button presses are required because he can simply select to play track 1 followed by track 2.

Given the number of tracks on Anya's favorite CD and the set of possible tracks for each song Anya wants played, determine the minimum number of button presses Arup must make to get a valid sequence of songs played. For the purposes of this problem, assume that at the very beginning the CD player is queued up to play track number 1, so that if the first song played is anything but track 1, some buttons will have to be pressed before the first song plays.

Currently, Anya's favorite CD is Divide by Ed Sheeran, and her favorite track numbers on this CD are 4, 5, and 7.

### Input

The first input line consists of two (space separated) positive integers: $t$ $(1 \le t \le 10^9)$, the number of tracks on Anya's favorite CD and s $(1 \le s \le 1000)$, the number of songs Anya would like to listen to from the CD. The next s input lines contain the possible track numbers for each song she would like to listen to. The $i^{th}$ of these lines starts with a positive integer, $n_i$ $(1 \le n_i \le 10)$, representing the number of choices Anya has provided for the $i^{th}$ song she listens to. This is followed by $n_i$ distinct positive integers, each in between 1 and $t$, inclusive, indicating the track numbers of the possible $i^{th}$ song Anya will listen to.

## Output

Output a single integer on a line by itself indicating the minimum number of button presses Arup can use to play the desired sequence of songs from the CD.

## Example

| Input | Output |
|---|---|
| 15 2 | 0 |
| 4 1 3 8 9 | |
| 2 12 2 | |
| 12 5 | 16 |
| 2 5 7 | |
| 2 5 7 | |
| 3 12 2 4 | |
| 1 9 | |
| 2 4 5 | |

## Problem F. Sub Matrix Sum

| | |
|---|---|
| Source file name: | sum.c, sum.cpp, sum.java, sum.py |
| Input: | `Standard` |
| Output: | `Standard` |

You have written many programs to search mazes so matrix search shouldn't be any different, or will it?

An integer matrix with R rows and C columns has $\binom{R}{2}\binom{C}{2}$ sub matrices. We want to select a sub matrix with sum (the sum of all integers in it) greater than or equal to a given integer $S$. We want the size of the sub matrix to be the least possible. The size of a sub matrix is defined as the number of elements in that sub matrix (i.e., number of rows $*$ number of columns in that sub matrix).

### Input

The first input line consists of three integers $R$, $C$ $(1 \leq R \leq 10^5;\ \ 1 \leq C \leq 10^5;\ \ 1 \leq R*C \leq 10^5)$ and $S$. Next $R$ lines contain the description of the matrix. Each of these $R$ lines contains $C$ integers separated by a single space. All integers (other than $R$ and $C$) are between $-10^9$ and $+10^9$ , inclusive.

### Output

Print the size of the minimum sub matrix whose sum is greater or equal to the given $S$. If there is no such sub matrix, output $-1$.

### Example

| Input | Output |
|---|---|
| `3 3 26`<br>`1 2 3`<br>`4 5 6`<br>`7 8 9` | `4` |
| `3 3 0`<br>`-1 -2 -3`<br>`-4 -5 -6`<br>`-7 -8 -9` | `-1` |
| `2 2 1`<br>`-1 -2`<br>`0 2` | `1` |

## Problem G. Mowing Mischief

| | |
|---|---|
| Source file name: | mowing.c, mowing.cpp, mowing.java, mowing.py |
| Input: | `Standard` |
| Output: | `Standard` |

Bessie's younger cousins, Ella and Bella, are visiting the farm. Unfortunately, they have been causing nothing but mischief since they arrived.

In their latest scheme, they have decided to mow as much grass as they can. The farm's prime grassland is in the shape of a large $t \times t$ square. The bottom-left corner is $(0,\ 0)$, and the top-right corner is $(t,\ t)$. The square therefore contains $(t+1)^2$ lattice points (points with integer coordinates).

Ella and Bella plan to both start at $(0,\ 0)$ and run at unit speed to $(t,\ t)$ while each holding one end of a very sharp and very stretchy wire. Grass in any area that is swept by this wire will be cut. Ella and Bella may take different paths, but each path consists of only upward and rightward steps, moving from lattice point to lattice point.

Bessie is rather concerned that too much grass will be cut, so she invents a clever plan to constrain the paths Ella and Bella take. There are $n$ yummy flowers scattered throughout the grassland, each on a distinct lattice point. Bessie will pick a set of $S$ flowers that will be required for both Ella and Bella to visit (so Ella's path must visit all the flowers in $S$, and so must Bella's path). In order to add as many waypoints to these paths as possible, Bessie will choose $S$ to be as large as possible among subsets of flowers that can be visited by a cow moving upward and rightward from $(0,\ 0)$ to $(t,\ t)$.

Ella and Bella will try to minimize the amount of grass they cut, subject to the restriction of visiting flowers in $S$. Please help Bessie choose $S$ so that the amount of grass cut is as small as possible.

### Input

The first input line contains two positive integers: $n$ ($1 \le n \le 2 \cdot 10^5$), indicating the number of yummy flowers scattered throughout the grassland and $t$ ($1 \le t \le 10^6$), where $(t,\ t)$ are the coordinates of the top right corner of the grid. Each of the next n lines contains the integer coordinates $(x_i,\ y_i)$ of a flower, with $1 \le x_i,\ y_i \le t-1$, for all $i$ ($1 \le i \le n$), and no two flowers lie on the same horizontal or vertical line.

### Output

Output a single integer on a line by itself giving the minimum possible amount of cut grass.

### Example

| Input | Output |
|---|---|
| 5 20<br>19 1<br>2 6<br>9 15<br>10 3<br>13 11 | 117 |

### Explanation

In this sample, it's best to pick the flowers at $(10,\ 3)$ and $(13,\ 11)$. With this selection, the first rectangle of grass cut has dimensions $10 \times 3$, the second one has $3 \times 8$, and the last one has the dimensions $7 \times 9$. The total area of these rectangles is $30 + 24 + 63 = 117$. A sub-optimal choice would be taking the flowers at $(2,\ 6)$ and $(9,\ 15)$. This choice cuts a rectangle of size $2 \times 6$, a second rectangle of size $7 \times 9$ and a third rectangle of size $11 \times 5$ for a total area of $12 + 63 + 55 = 130$, which is greater than 117. The third possible choice which is sub-optimal would be to pick the flower at $(2,\ 6)$ and the flower at $(13,\ 11)$, which results in a total area of grass cut of 152 units$^2$.

## Problem H. Rounding Many Ways

| | |
|---|---|
| Source file name: | rounding.c, rounding.cpp, rounding.java, rounding.py |
| Input: | `Standard` |
| Output: | `Standard` |

Timothy and Alex's hopes and dreams of running for UCF's Student Government Association have been crushed by the realization that their campaign ticket would not be alliterative. So, they have decided to analyze statistics from many different polls given to students to determine what pair of programming team members would be best situated to win the election. However, there is a problem. All of the statistics have been rounded off. This would not be an issue apart from the fact that the pollsters forgot to mention how the number was rounded!
(Math Terminology Note: if we round, say, 198 to 200, then 198 is called the "true value" and 200 is called the "rounded value".)

For example, the rounded value 750 could have come from a true value rounded to the nearest 10 or maybe even the nearest 250. It may also have come from a true value rounded to the nearest 1 (thus not rounding it at all). Thus, the true value could have been something like 625 or maybe much closer like 748.

Luckily for Timothy and Alex, after some reconnaissance work, they have discovered the general rounding methods used:

- The original statistic was a positive integer $S$
- Then a positive integer $X$ was chosen such that $X$ is a divisor of some power of 10, i.e., there exist non-negative integers $Y$ and $Z$ such that $X \cdot Y = 10^Z$
- Finally the statistic $S$ was rounded to the nearest positive multiple of $X$ to get the rounded value $N$, i.e., there exists a positive integer $W$ such that $X \cdot W = N$ and $|S\breve{\ }N|$ is minimized.

Given the rounded value, find all the different ways it could have been rounded (derived). In other words, given $N$ and using the above constraints, you are to find all values of $X$ that satisfy both of the following two equations:

- $X \cdot Y = 10^Z$
- $X \cdot W = N$

### Input

The first and only line of input contains an integer, $N$ ($1 \leq N \leq 10^{18}$), representing the rounded value.

### Output

First print out a single line containing an integer representing the number of different $X$ values that the rounded value $N$ could have been derived from. Then print out all of these values of $X$, in increasing order, on separate lines.

## Example

| Input | Output |
|---|---|
| 30 | 4<br>1<br>2<br>5<br>10 |
| 120 | 8<br>1<br>2<br>4<br>5<br>8<br>10<br>20<br>40 |
| 8 | 4<br>1<br>2<br>4<br>8 |

## Explanation

**Explanation for the first Example Input/Output:**

**Output: 1**
$X = 1$, $Y = 10$, $Z = 1$, $W = 30$
Rounded to nearest multiple of 1: $1 \cdot 10 = 10$, $1 \cdot 30 = 30$

**Output: 2**
$X = 2$, $Y = 5$, $Z = 1$, $W = 15$
Rounded to nearest multiple of 2: $2 \cdot 5 = 10$, $2 \cdot 15 = 30$

**Output: 5**
$X = 5$, $Y = 2$, $Z = 1$, $W = 6$
Rounded to nearest multiple of 5: $5 \cdot 2 = 10$, $5 \cdot 6 = 30$

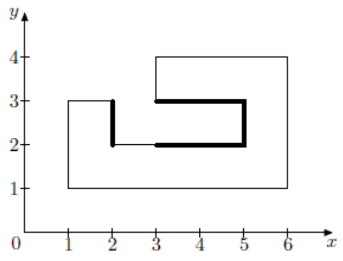
**Output: 10**
$X = 10$, $Y = 1$, $Z = 1$, $W = 3$
Rounded to nearest multiple of 10: $10 \cdot 1 = 10$, $10 \cdot 3 = 30$

## Problem I. Intelligence in Perpendicularia

| | |
|---|---|
| Source file name: | intel.c, intel.cpp, intel.java, intel.py |
| Input: | `Standard` |
| Output: | `Standard` |

There are only two directions in Perpendicularia: vertical and horizontal. Perpendicularia government are going to build a new secret service facility. They have some proposed facility plans and want to calculate total secured perimeter for each of them.

The total secured perimeter is calculated as the total length of the facility walls invisible for the perpendicularly-looking outside observer. The figure below shows one of the proposed plans and corresponding secured perimeter.



Write a program that calculates the total secured perimeter for the given plan of the secret service facility.

### Input

The plan of the secret service facility is specified as a polygon.

The first line of the input contains one integer $n$ — the number of vertices of the polygon ($4 \le n \le 1000$). Each of the following $n$ lines contains two integers $x_i$ and $y_i$ – the coordinates of the $i$-th vertex ($-10^6 \le x_i, y_i \le 10^6$). Vertices are listed in the consecutive order.

All polygon vertices are distinct and none of them lie at the polygon's edge. All polygon edges are either vertical ($x_i = x_{i+1}$) or horizontal ($y_i = y_{i+1}$) and none of them intersect each other.

### Output

Output a single integer — the total secured perimeter of the secret service facility.

### Example

| Input | Output |
|---|---|
| 10<br>1 1<br>6 1<br>6 4<br>3 4<br>3 3<br>5 3<br>5 2<br>2 2<br>2 3<br>1 3 | 6 |

## Problem J. SGA President

| | |
|---|---|
| Source file name: | sga.c, sga.cpp, sga.java, sga.py |
| Input: | `Standard` |
| Output: | `Standard` |

After an amazing performance at World Finals, Timothy and Alex, who are no longer eligible for ICPC, have decided to look for a new challenge. They'd like to run for SGA President and Vice President. Unfortunately, they have realized that with tickets from the previous election such as Josh/Jad and Brad/Breon, they have no hope of winning because all winning tickets must have two distinct names that start with the same first letter, so Timothy and Alex just won't do.

Naturally, Timothy was despondent about this revelation and to make himself feel better came up with a problem for locals. Given the names of each student at UCF, Timothy wondered how many potential winning pairs for SGA President and Vice-President there might be. In order for a pair to have the potential to win, their names must be different but start with the same first letter. Since President and Vice President are different roles, we count the ticket of Josh and Jad differently than the ticket of Jad and Josh. (The first name listed is the candidate for President while the second name listed is the corresponding candidate for Vice President.) Note that UCF has many students that share a first name, so there might be several potential winning pairs of Josh and Jad. For example, if there are 10 Joshes and 3 Jads on campus, there are 30 Josh/Jad pairs with a Josh running for President and a Jad running for Vice President (and these should all be counted).

Given the names of each UCF student, calculate the number of possible President/Vice-President pairs who have a potential to win the SGA election.

### Input

The first line of input contains a single positive integer, $n$ ($n \leq 66183$), representing the number of UCF students. The following $n$ lines each contain a single first name of one UCF student. All names will consist of uppercase letters only and be between 1 and 20 letters long, inclusive. Each line represents a distinct student, but distinct students may have the same first name.

### Output

On a line by itself, output the total number of President-Vice President pairs that have a chance to win the SGA election.

## Example

| Input | Output |
|---|---|
| 10<br>JOSH<br>JAD<br>JENNIFER<br>JENNIFER<br>JALEN<br>HASAAN<br>ALI<br>TIM<br>ALEX<br>TRAVIS | 22 |
| 5<br>ALEX<br>BRANDY<br>CELINE<br>DWAYNE<br>ELIZABETH | 0 |