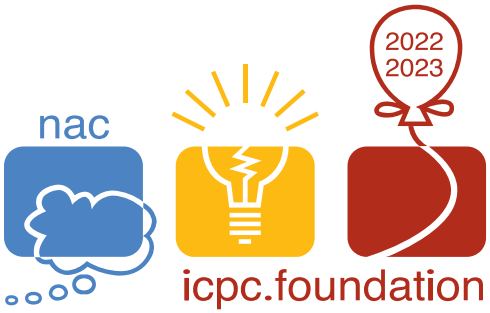
icpc international collegiate programming contest

ICPC North America Contests

Pacific Northwest Regional Contest

Official Problem Set



Pacific Northwest Regional Programming Contest
Division 2
25 February 2023

- The languages supported are C, C++ 17 (with Gnu extensions), Java, Python 3 (with pypy3), and Kotlin.
- Python 2 and C# are not supported this year.
- For all problems, read the input data from standard input and write the results to standard output.
- In general, when there is more than one integer or word on an input line, they will be separated from each other by exactly one space. No input lines will have leading or trailing spaces, and tabs will never appear in any input.
- Submit only a single source file for each problem.
- Python may not have sufficient performance for many of the problems; use it at your discretion.

Problem A
Three Dice

Time Limit: 5 sec

Given a list of three-letter words, generate one possible set of three, six-sided dice such that each word can be formed by the top faces of some arrangement of the three dice. You must distribute 18 distinct letters across the 18 total faces of the dice. There may be multiple possible sets of dice that satisfy the requirement; any correct set will be accepted.

Input

The first line of input contains an integer n ($1 \leq n \leq 1{,}000$), which is the number of words.

Each of the next n lines contains one three-letter word made up only of lowercase letters (`a`–`z`). There may be duplicate words in the list, and the words might contain identical letters.

Output

Output a single line. If there exists a set of dice that can form all of the words, output any such set. Output the set of dice as one line with three space-separated strings, each consisting of six lowercase letters. If no such set of dice can be formed, output a single line with the number 0.

Sample Input 1	Sample Output 1
3 lad fin sly	zounds plight fakery

Sample Input 2	Sample Output 2
1 dad	0

Sample Input 3	Sample Output 3
11 aft cog far irk kit yes tau rag own uke via	vortex whacky fusing

Problem B
Alchemy

Time Limit: 1 sec

You just finished day one of your alchemy class! For your alchemy homework, you have been given a string of lowercase letters and wish to make it a palindrome. You're only a beginner at alchemy though, so your powers are limited. In a single operation, you may choose exactly two adjacent letters and change each of them into a different lowercase letter. The resulting characters may be the same as or different from one another, so long as they were both changed by the operation.

Formally, if the string before the operation is s and you chose to change characters s_i and s_{i+1} to produce string t, then $s_i \neq t_i$ and $s_{i+1} \neq t_{i+1}$ must be true, but $t_i = t_{i+1}$ is permitted.

Compute the minimum number of operations needed to make the string a palindrome.

Input

The single line of input contains a string of n $(2 \leq n \leq 100)$ lowercase letters, the string you are converting into a palindrome.

Output

Output a single integer, which is the minimum number of operations needed to make the string a palindrome.

Sample Input 1	Sample Output 1
ioi	0

Sample Input 2	Sample Output 2
noi	1

Sample Input 3	Sample Output 3
ctsc	1

Sample Input 4	Sample Output 4
fool	2

Sample Input 5	Sample Output 5
vetted	2

Problem C
Champernowne Count

Time Limit: 1 sec

The nth Champernowne word is obtained by writing down the first n positive integers and concatenating them together. For example, the 10th Champernowne word is "12345678910".

Given two positive integers n and k, count how many of the first n Champernowne words are divisible by k.

Input

The single line of input contains two integers, n ($1 \leq n \leq 10^5$) and k ($1 \leq k \leq 10^9$).

Output

Output a single integer, which is a count of the first n Champernowne words divisible by k.

Sample Input 1	Sample Output 1
4 2	2

Sample Input 2	Sample Output 2
100 7	14

Sample Input 3	Sample Output 3
314 159	4

Sample Input 4	Sample Output 4
100000 999809848	1

This page is intentionally left blank.

Problem D
Hunt the Wumpus

Time Limit: 1 sec

Hunt the Wumpus is a game you play against the computer on a 10×10 grid. At the beginning of the game, four wumpuses are randomly placed on the grid (with no two sharing a location). You need to find and capture all four wumpuses. You guess a square by entering the coordinates as two decimal digits. If you correctly guess the location of a wumpus, you are told you did so, and that wumpus is captured and removed from the grid. Whether or not you hit a wumpus, the Manhattan distance between your guess and the closest wumpus is reported to you. You can use this to locate and find each wumpus. The game ends when the last wumpus is captured.

You have been asked to write the computer portion of the game. User guesses and randomly-generated wumpus locations are defined by a two-digit number, where the high digit is x and the low digit is y for the point (x, y).

To make the game deterministic, we will use our own pseudo-random-number generator, with a supplied seed s. Each random number is generated by first setting $s \leftarrow s + \text{floor}(s/13) + 15$ and then returning the two low digits of s. The first four distinct numbers generated by this process are the locations of the four wumpuses. For a seed of 132, the first location is given by the two low digits of $132 + 10 + 15$, which is 57 and corresponds to the position $(5, 7)$.

Input

The first line contains a single integer s ($10^4 \leq s \leq 10^6$), the seed for the random number generator. Each of the remaining lines contains a two-digit number (possibly with leading zeros). These are the guesses that a player has made, each corresponding to a single grid location.

The input will always be well-formed, and will describe a complete game. The last user guess will always find the last wumpus. There will be at most 250 guesses in any input.

Output

Output the computer's response for each player guess. If a guess hits a wumpus, you should output "`You hit a wumpus!`" on a line. Whether or not the player hit a wumpus, if any wumpuses remain uncaptured, output the Manhattan distance to the closest remaining wumpus. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$, and will therefore always be an integer.

At the end of the game, report the total number of moves m required to locate all the wumpuses by outputting "`Your score is` m `moves.`" on a line.

Sample Input 1	Sample Output 1
203811 00 01 02 03	You hit a wumpus! 1 You hit a wumpus! 1 You hit a wumpus! 1 You hit a wumpus! Your score is 4 moves.

Sample Input 2	Sample Output 2
101628 00 40 60 68 78 95	4 You hit a wumpus! 2 You hit a wumpus! 8 1 You hit a wumpus! 5 You hit a wumpus! Your score is 6 moves.

Problem E
Color Tubes

Time Limit: 1 sec

There is a new puzzle generating buzz on social media—Color Tubes. The rules are relatively simple: you are given $n + 1$ tubes filled with $3n$ colored balls. Each tube can hold at most 3 balls, and each color appears on exactly 3 balls (so there are n colors).

Using a series of moves, you are supposed to reach a Color Tubes state—each tube should either hold balls of a single color or it should be empty.

The only move allowed is to take the top ball from one tube and place it into a different tube that has room for it (i.e. holds at most two balls before the move).

You want to write a program to solve this puzzle for you. Initially, you are not interested in an optimal solution, but you want your program to be good enough to solve any puzzle configuration using at most $20n$ moves.

Input

The first line of input contains a single integer n ($1 \leq n \leq 1{,}000$), which is the number of colors.

Each of the next $n + 1$ lines contains three integers b, m and t ($0 \leq b, m, t \leq n$), which are the descriptions of each tube, where b is the color of the ball on the bottom, m is the color of the ball in the middle, and t is the color of the ball on the top.

The tubes are numbered from 1 to $n + 1$ and are listed in order. The colors are numbered from 1 to n. The number 0 describes an empty space. It is guaranteed that no empty space will be below a colored ball.

Output

On the first line output an integer m, the number of moves that your program will use to solve the puzzle. Remember, m has to be at most $20n$.

On the next m lines, output two space-separated integers u and v that describe a move ($1 \leq u, v \leq n + 1$). In each move, you are taking the uppermost ball out of tube u and placing it in tube v, where it will fall until it hits the uppermost ball already in that tube, or the bottom of the tube if the tube is empty.

Your solution will be deemed incorrect if it uses more than $20n$ moves, or any of the moves are not allowed, or the final configuration is not a Color Tubes state.

Sample Input 1

```
3
2 2 0
1 3 1
3 1 2
3 0 0
```

Sample Output 1

```
6
3 1
2 3
2 4
3 2
3 2
3 4
```

Sample Input 2

```
1
0 0 0
1 1 1
```

Sample Output 2

```
0
```

Problem F
Food Processor

Time Limit: 2 sec

You have a food processor with a variety of blades that can be attached to it, as well as some food you would like to process into smaller pieces.

The food processor can have one blade attached at any time. Each blade processes food by reducing its average piece size at a particular exponential rate, but it also has a maximum average piece size requirement; if the average piece size of the food is too big for the blade, the food processor will get stuck. Given a starting average food piece size, a target average piece size, and a set of blades for your food processor, determine the minimum amount of processing time needed to process your food into the target average piece size.

Note that we only care about the time spent actively processing food; we do not track time spent switching out blades or loading/unloading the food processor.

Input

The first line of input contains three integers s, t, and n ($1 \leq t < s \leq 10^6, 1 \leq n \leq 10^5$), where s is the starting average piece size, t is the target average piece size, and n is the number of blades.

Each of the next n lines contains two integers m and h ($1 \leq m, h \leq 10^6$). These are the blades, where m is the maximum average piece size of the blade and h is the number of seconds the blade needs to halve the average piece size.

Output

Output a single number, which is the minimum amount of time in seconds needed to process the food to the target average piece size. If it is not possible to reach the target, output -1. Your answer should have a *relative* error of at most 10^{-5}.

Sample Input 1	Sample Output 1
10 1 2 10 10 4 5	23.219281

Sample Input 2	Sample Output 2
10000 9999 1 10000 1	1.4427671804501932E-4

Problem G
Fading Wind

Time Limit: 1 sec

You're competing in an outdoor paper airplane flying contest, and you want to predict how far your paper airplane will fly. Your design has a fixed factor k, such that if the airplane's velocity is at least k, it will rise. If its velocity is less than k it will descend.

Here is how your paper airplane will fly:

- You start by throwing your paper airplane with a horizontal velocity of v at a height of h. There is an external wind blowing with a strength of s.
- While $h > 0$, repeat the following sequence:
  - Increase v by s. Then, decrease v by $\max(1, \lfloor \frac{v}{10} \rfloor)$. Note that $\lfloor \frac{v}{10} \rfloor$ is the value of $\frac{v}{10}$, rounded down to the nearest integer if it is not an integer.
  - If $v \geq k$, increase h by one.
  - If $0 < v < k$, decrease h by one. If h is zero after the decrease, set v to zero.
  - If $v \leq 0$, set h to zero and v to zero.
  - Your airplane now travels horizontally by v units.
  - If $s > 0$, decrease it by 1.

Compute how far the paper airplane travels horizontally.

Input

The single line of input contains four integers h, k, v, and s ($1 \leq h, k, v, s \leq 10^3$), where h is your starting height, k is your fixed factor, v is your starting velocity, and s is the strength of the wind.

Output

Output a single integer, which is the distance your airplane travels horizontally. It can be shown that this distance is always an integer.

Sample Input 1	Sample Output 1
1 1 1 1	1

Sample Input 2	Sample Output 2
2 2 2 2	9

Sample Input 3	Sample Output 3
1 2 3 4	68

Sample Input 4	Sample Output 4
314 159 265 358	581062

Problem H
Creative Accounting

Time Limit: 5 sec

When accounting for the profit of a business, we can divide consecutive days into fixed-sized segments and calculate each segment's profit as the sum of all its daily profits. For example, we could choose seven-day segments to do our accounting in terms of weekly profit. We also have the flexibility of choosing a segment's starting day. For example, for weekly profit we can start a week on a Sunday, Monday, or even Wednesday. Choosing different segment starting days may sometimes change how the profit looks on the books, making it more (or less) attractive to investors.

As an example, we can divide ten consecutive days of profit (or loss, which we denote as negative profit) into three-day segments as such:

$$3, 2, -7 \mid 5, 4, 1 \mid 3, 0, -3 \mid 5$$

This gives us four segments with profit $-2, 10, 0, 5$. For the purpose of this division, partial segments with fewer than the fixed segment size are allowed at the beginning and at the end. We say a segment is profitable if it has a strictly positive profit. In the above example, only two out of the four segments are profitable.

If we try a different starting day, we can obtain:

$$3, 2 \mid -7, 5, 4 \mid 1, 3, 0 \mid -3, 5$$

This gives us four segments with profit $5, 2, 4, 2$. All four segments are profitable, which makes our business look much more consistent.

You're given a list of consecutive days of profit, as well as an integer range. If we can choose any segment size within that range and any starting day for our accounting, what is the minimum and maximum number of profitable segments that we can have?

Input

The first line of input has three space-separated integers n, ℓ and h ($1 \leq \ell \leq h \leq n \leq 3 \times 10^4$, $h - \ell \leq 1{,}000$), where n is the number of days in the books, ℓ is the minimum possible choice of segment size, and h is the maximum possible choice of segment size.

Each of the next n lines contains a single integer p ($-10^4 \leq p \leq 10^4$). These are the daily profits, in order.

Output

Output on a single line two space-separated integers min and max, where min is the minimum number of profitable segments possible, and max is the maximum number of profitable segments possible. Both min and max are taken over all possible choices of segment size between ℓ and h and all possible choices of starting day.

Sample Input 1	Sample Output 1
10 3 5 3 2 -7 5 4 1 3 0 -3 5	2 4

Problem I
I Could Have Won

Time Limit: 1 sec

"We will be closing in about 5 minutes. Thank you for visiting the ICPC gym today."

With this announcement, Alice and Bob stopped playing their rock-paper-scissors marathon in the middle of the 10th game. Each player scores a point if their throw beats the other player's throw. Each game was played by the *first-to-11* rule, meaning that whoever scores 11 points first wins the game. Today, Bob narrowly defeated Alice by a single game; he scored 11 points first in five games, while Alice only scored 11 points first in four games.

After carefully inspecting how each game was played, however, Alice realized that she could have won more games than Bob if they played under slightly different rules, such as *first-to-5* or *first-to-8*, instead of the regular *first-to-11*.

Given the sequence of points scored by Alice and Bob, determine all values of k such that Alice would have won more games than Bob under the *first-to-k* rule.

Both Alice and Bob start with zero points at the beginning of a game. As soon as one player reaches k points, that player wins the game, and a new game starts. Alice wins a game if she scores k points before Bob does. Neither player wins the game if it's interrupted by the gym closing before either player reaches k points.

Input

The single line of input consists of a string of uppercase letters "A" or "B", denoting who scored each point from the beginning of the rock-paper-scissors marathon. The length of the string is between 1 and 2,000 letters, inclusive. "A" means Alice scored the point, "B" means Bob scored the point.

Output

On the first line, output the number of positive integers k for which a *first-to-k* rule would have made Alice win more games than Bob. If this number isn't zero, on the next line output all such values of k in increasing order, separated by spaces.

Sample Input 1	Sample Output 1
BBAAABABBAAABB	3 3 6 7

Sample Input 2	Sample Output 2
AABBBAAB	2 2 4

Problem J
Sun and Moon

Time Limit: 1 sec

You recently missed an eclipse and are waiting for the next one! To see any eclipse from your home, the sun and the moon must be in alignment at specific positions. You know how many years ago the sun was in the right position, and how many years it takes for it to get back to that position. You know the same for the moon. When will you see the next eclipse?

Input

The input consists of two lines.

The first line contains two integers, d_s and y_s $(0 \leq d_s < y_s \leq 50)$, where d_s is how many years ago the sun was in the right position, and y_s is how many years it takes for the sun to be back in that position.

The second line contains two integers, d_m and y_m $(0 \leq d_m < y_m \leq 50)$, where d_m is how many years ago the moon was in the right position, and y_m is how many years it takes for the moon to be back in that position.

Output

Output a single integer, the number of years until the next eclipse. The data will be set in such a way that there is not an eclipse happening right now and there will be an eclipse within the next $5{,}000$ years.

Sample Input 1	Sample Output 1
3 10 1 2	7

This page is intentionally left blank.

Problem K
Chocolate Chip Fabrication

Time Limit: 1 sec

You are making a chocolate chip cookie using a machine that has a rectangular pan composed of unit squares. You have determined the shape of your cookie, which occupies some squares in that area. Each square of your cookie must be chocolate chipified.

To make the cookie you will repeatedly perform the following two steps:

1. You place cookie dough in some unit squares.
2. You expose the cookie dough to a shallow chocolate chip solution. Any cookie dough square that does not have all four adjacent squares (up, down, left, right) filled with cookie dough becomes chocolate chipified. Note that any cookie dough in a square on the boundary of the pan always gets chipified.

The following example shows how to make a cookie of the shape shown on the left (s):

```
(s)     (a1)    (a2)    (b1)    (b2)
-X-X-   -D-D-   -C-C-   -C-C-   -C-C-
XXXXX   -D-D-   -C-C-   DCDCD   CCCCC
XXXXX   -DDD-   -CCC-   DCCCD   CCCCC
-XXX-   --D--   --C--   -DCD-   -CCC-
--X--   -----   -----   --D--   --C--
```

First you place cookie dough in 8 squares (a1). All squares become chipified after the first solution exposure (a2). You place cookie dough in 8 more squares (b1). The second exposure makes every square chipified and completes the cookie (b2).

Your chocolate chip solution is expensive, so you want to ensure that you perform the exposure as few times as possible. Given a cookie shape, determine the minimum number of chocolate chip solution exposures required to make the cookie.

Input

The first line of input contains two integers n and m ($1 \leq n, m \leq 1{,}000$), indicating the pan has n rows and m columns of unit squares.

Each of the next n lines contains a string of exactly m characters, where each character is either "X", representing a square occupied by your cookie, or "–", representing an empty square.

The shape of your cookie occupies at least one square. Note that the shape may consist of multiple pieces that are disconnected.

Output

Output the minimum number of chocolate chip solution exposures required to make your cookie.

Sample Input 1	Sample Output 1
5 5 -X-X- XXXXX XXXXX -XXX- --X--	2

Sample Input 2	Sample Output 2
4 5 --XXX --X-X X-XXX XX---	1

Sample Input 3	Sample Output 3
5 5 XXXXX XXXXX XXXXX XXXXX XXXXX	3

Problem L
Distinct Parity Excess

Time Limit: 3 sec

A property of any positive integer is its *prime parity*, which is derived from the count of its distinct prime factors. If this count is even, the prime parity is even; if the count is odd, the prime parity is odd.

You are given a sequence of ranges to test. Each range is given as two numbers a and b, defining the range from a to b inclusive. You want to compute the excess of even parity integers over odd parity integers over this range. If there are more odd parity integers, the computed difference will be negative.

Input

The first line of the input contains a single integer n $(1 \leq n \leq 100)$, which is the number of ranges to test.

Each of the next n lines contains two integers a and b $(2 \leq a \leq b \leq 10^7)$, which is a range to test.

Output

Output n lines, one for each range in the input. For each range, output a single integer giving the excess of even parity integers over odd parity integers.

Sample Input 1	Sample Output 1
3 2 2 2 5 2 10	-1 -4 -5

Sample Input 2	Sample Output 2
8 2 100 2 50 50 100 2 1000 100 143 2 1000000 80000 90000 1000000 1000000	13 -1 15 63 0 -1909 -31 1

Problem M
Restaurant Opening

Time Limit: 1 sec

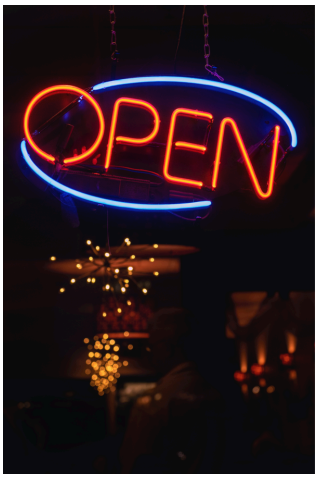


Image by wirestock on Freepik

It is said that the three most important factors for determining whether or not a business will be successful are location, location, and location. The Incredible Cooks Preparing Cuisine are opening a new restaurant in the International City Promoting Cooking, and they have hired you to find the optimal location for their restaurant.

You decide to model the city as a grid, with each grid square having a specified number of people living in it. The distance between two grid squares (r_1, c_1) and (r_2, c_2) is $|r_1 - r_2| + |c_1 - c_2|$. In order to visit the restaurant, each potential customer would incur a cost equal to the minimum distance from the grid square in which they live to the grid square containing the proposed location of the restaurant. The total cost for a given restaurant location is defined as the sum of the costs of everyone living in the city to visit the restaurant.

Given the current city layout, compute the minimum total cost if the Incredible Cooks Preparing Cuisine select their next restaurant location optimally.

Input

The first line of input contains two integers, n and m ($1 < n, m \leq 50$), where n is the number of rows in the city grid and m is the number of columns.

Each of the next n lines contains m integers g_{ij} ($0 \leq g_{ij} \leq 50$), which specifies the number of people living in the grid square at row i, column j.

Output

Output a single integer, which is the total cost if the restaurant is selected optimally.

Sample Input 1	Sample Output 1
2 2 1 2 3 4	7

Sample Input 2	Sample Output 2
1 10 3 49 4 31 10 31 50 24 10 42	591