ICPC SOUTH PACIFIC DIVISIONALS

SEPTEMBER 21, 2019

Contest Problems

This contest contains thirteen problems over 30 pages. Good luck.

For problems that state "*Your answer should have an absolute or relative error of less than* 10^{-9}", your answer, x, will be compared to the correct answer, y. If $|x - y| < 10^{-9}$ or $\frac{|x-y|}{|y|} < 10^{-9}$, then your answer will be considered correct.

Definition 1

For problems that ask for a result modulo m:
If the correct answer to the problem is the integer b, then you should display the unique value a such that:

- $0 \leq a < m$
  and
- $(a - b)$ is a multiple of m.

Definition 2

A string $s_1 s_2 \cdots s_n$ is lexicographically smaller than $t_1 t_2 \cdots t_\ell$ if

- there exists $k \leq \min(n, \ell)$ such that $s_i = t_i$ for all $1 \leq i < k$ and $s_k < t_k$
  or
- $s_i = t_i$ for all $1 \leq i \leq \min(n, \ell)$ and $n < \ell$.

Definition 3

- Uppercase letters are the uppercase English letters $(A, B, \ldots, Z)$.
- Lowercase letters are the lowercase English letters $(a, b, \ldots, z)$.

Definition 4

Unless otherwise specified, the distance between two points (x_0, y_0) and (x_1, y_1) is defined as its Euclidean distance:

$$\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}.$$

Problem A
Array Index
Time limit: 1 second

Many programming languages include the concept of the *array*: a sequential collection of items of the same data type, each of which can be accessed through the integer *index* of its location in the array.

There is an old joke that as some programming languages have arrays with indices that start from 0 and others have arrays with indices that start from 1, it is fortunate that no language uses the compromise of having arrays with indices that start from 0.5. This problem is based on such a “compromised” language.

The only valid array names in the language are single, uppercase letters. As in C, C++, and Java, array indexing is indicated with square brackets and array indices increase in steps of 1. For example, `A[0.5]` refers to the first location in the array `A`, while `Z[3.5]` refers to the fourth location in the array `Z`.

Arrays only contain integer values, are automatically created when first used in a program, and automatically expand as necessary to include values at any indices referred to. For example, if `Z[3.5]` is the first location of `Z` mentioned, then `Z` is created with four locations, (`Z[0.5]`, ..., `Z[3.5]`). The integer value of an array location is always initialised to 0 when that location comes into existence, whether this is upon the creation of an array or the expansion of an array.

In this problem, the only type of statement in the language to be considered is a simple use of the += operation, which adds a constant integer value on its right-hand side to the value at the array location (indexed by a constant value) on its left-hand side. For example, if a program starts with the following two statements:

```
Z[3.5] += 1
Z[3.5] += 3
```

the value in `Z[3.5]` after these statements are executed is 4, and the values in all other locations in `Z` are 0.

Given a program of this form, what are the names of all the arrays used in the program and the non-zero values in arrays after the program has executed?

Input

The input contains a single program in the programming language. The program contains L ($1 \leq L \leq 100$) valid statements of the form described above, one statement per line. Each statement contains 3 items: the identifier of the array location, the += operator and the integer V ($-100 \leq V \leq 100$), which is the value to be added to the value at the array location. The identifier of the array location consists of: the uppercase letter which is the name of the array, the `[`, the index value, and the `]`. The index value is some integer k ($0 \leq k < 10^7$) followed by `.5`.

Output

For each array mentioned in the program, (in ascending alphabetical order of array name): first display the name of the array on a line by itself, then, in ascending numerical order of index value, print any non-zero values in that array each on a line by itself.

Sample Input 1	Sample Output 1

`Z[3.5] += 1`
`Z[3.5] += 3`
	`Z`
	`4`

Sample Input 2

```
A[0.5] += -2
A[1.5] += 3
A[2.5] += 3
Z[3.5] += 1
Z[44.5] += 3
```

Sample Output 2

```
A
-2
3
3
Z
1
3
```

Sample Input 3

```
A[0.5] += 0
```

Sample Output 3

```
A
```

Sample Input 4

```
Y[1.5] += -100
Y[0.5] += -100
Y[1.5] += 100
Y[2.5] += -50
Y[0.5] += 99
Y[11.5] += -20
```

Sample Output 4

```
Y
-1
-50
-20
```

Problem B
Binary Flips
Time limit: 1 second

A photo of Erica

Erica has recently grown fond of large binary strings (strings consisting of only the digits `0` and `1`). Specifically, binary strings known as *binary flips* have caught her eye. A binary flip is a string of length at least 3 and with odd length that alternates between `1` and `0` (starting and ending with `1`).

`101`, `10101`, `1010101`, `101010101`, etc.

Erica has already written down a binary string, but she wants to turn it into a binary flip. She can turn a binary string into a binary flip by removing some of the characters (possibly none and not necessarily contiguous) from the binary string so that the remaining characters form a binary flip.

For example, if Erica's binary string is `10101`, there are 5 ways to form a binary flip: (a) remove nothing (leaving `10101`), (b) remove the first and second characters (leaving `101`), (c) remove the second and third characters (leaving `101`), (d) remove the third and fourth characters (leaving `101`), and (e) remove the fourth and fifth characters (leaving `101`).

Given a binary string, count the number of sets of characters that Erica could remove to create a binary flip.

Input

The input comprises a single line containing one of Erica's binary strings. The number of characters in the binary string is between 1 and 100 000, inclusive.

Output

Display the number of sets of characters that Erica could remove to create a binary flip, modulo 1 000 000 007.

Sample Input 1	Sample Output 1
`101`	`1`

Sample Input 2	Sample Output 2
`11011`	`4`

Sample Input 3	Sample Output 3
`10101`	`5`

This page is intentionally left (almost) blank.

Problem C
Codenames
Time limit: 4 seconds

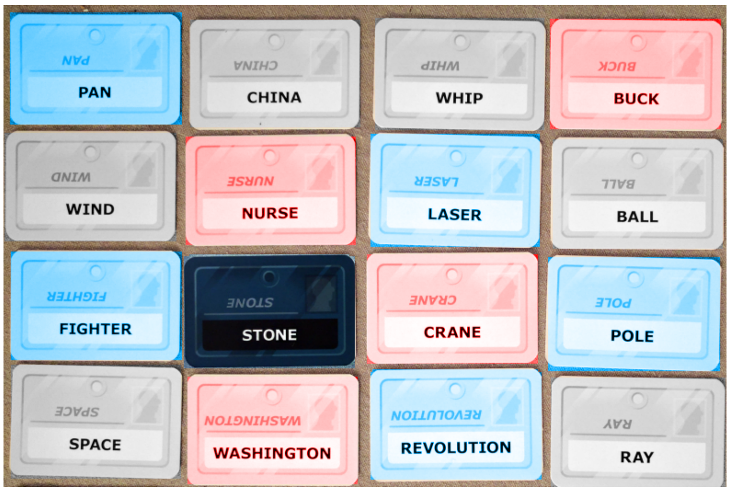
Alice and Bob are playing yet another game. This time, they are playing with some friends. They are playing a special variant of the board game *Codenames* that they have invented, which they use to settle arguments about what game to play next. In the game, Alice and Bob have to provide hints to their teams so that they can correctly guess the identities of spies. Each spy has a codename, hence the name of the game.

Their variant of Codenames works as follows. Alice and Bob are both *codemasters*. Each codemaster has their own team of *guessers*. The game is played on a 4×4 square grid of words. We will refer to these words as the *codenames*. Every codename has a special type that is hidden from the guessers but known to the codermasters. A codename's type can be either blue, which means that it belongs to Alice, red, which means that it belongs to Bob, the assassin, which means it belongs to no codemaster and should not be guessed, or neutral, which means it belongs to no codemaster and can be safely guessed.

The codemasters will take turns providing hints to their guessers. Alice takes the first turn. In a turn, the codemaster must select a hint from a predefined list of hints. The guessers will then guess the hint's associated codenames in lexicographical order, skipping over any codenames that were guessed in a previous turn.

If they guess a codename that is not the type associated with their codemaster, they must stop guessing. If they guess the assassin, then their team immediately loses. If the last codename of some codemaster's type (either blue or red) is guessed, then the corresponding team immediately wins the game no matter which team guessed it. If a codemaster says a hint and their guessers cannot guess any cards (because all the associated codenames are already guessed), then their guessers become stumped, and their team loses.

For example, consider the following grid of codenames:



Adapted from: Best Play. The codename "STONE" is the assassin.

Suppose that the hint `big` is associated with, in lexicographical order, `crane`, `space`, and `stone`. If Bob, who is on the red team, gave the clue, his team starts guessing with `crane`, then `space`, and then they will stop guessing since they guessed a card of the wrong type. It would then be Alice's turn to give a clue. However, if `space` was guessed in a previous turn, then Bob's guessers would guess `crane`, then `stone`, after which the red team will lose immediately because they guessed the assassin. If `crane` was the last red codename remaining (no matter who gave the clue), the red team will win immediately after `crane` is guessed. Furthermore, if Alice gave the hint `bird` which is associated only with `crane` and if that was already guessed in a previous turn, her guessers would become stumped and Alice's team would lose.

Note that hints may be reused as many times. If Alice and Bob both play optimally, who wins?

Input

The first four lines of input define the board of codenames. Each line contains four words. Each word contains between 1 and 10 lowercase letters. All the codenames are guaranteed to be unique.

Following this are four lines. Each line contains four letters. These define the types of codenames. The character `B` represents blue, `R` represents red, `A` represents the assassin, and `.` represents neutral. There are exactly 5 blue codenames, exactly 4 red codenames, and exactly 1 assassin codename.

The next line contains a single integer H ($1 \leq H \leq 50$), which is the number of predefined hints. The next H lines define the hints. Each of these line starts with a word, which is the hint, containing between 1 and 10 lowercase letters. Following this on the line is an integer C ($1 \leq C \leq 10$), which is the number of codenames associated with this hint. Following this on the line are C words. Each such word will be a codename that appears on the board.

Output

Display `Alice` if Alice wins, otherwise, display `Bob`.

Sample Input 1

```
pan china whip buck
wind nurse laser ball
fighter stone crane pole
space washington revolution ray
B..R
.RB.
BARB
.RB.
4
war 3 space revolution fighter
big 3 crane space stone
www 3 wind whip washington
bird 1 crane
```

Sample Output 1

```
Alice
```

Sample Input 2

```
a b c d
e f g h
i j k l
m n o p
BBBB
BRRR
R..A
....
1
hint 5 a b c d e
```

Sample Output 2

```
Alice
```

Sample Input 3

```
a b c d
e f g h
i j k l
m n o p
BBBB
BRRR
R..A
....
3
hinta 5 l f g h i
hintb 2 a b
hintc 2 c d
```

Sample Output 3

```
Bob
```

This page is intentionally left (almost) blank.

Problem D
Dodging Spoilers
Time limit: 5 seconds

There is a tournament that consists of 2^N players (numbered from 1 to 2^N) who play in a total of N rounds. During each round, all the players are paired off into matches. Every player who loses their match is eliminated and removed from the tournament (there are no ties). Every player who wins their match moves onto the next round and plays another winner from the preceding round until there is only one player left.

The judges have given you a list of which pairs of players were in each of the $2^N - 1$ matches, but not their outcomes nor which round they were played in (since you would like to avoid spoilers).

It is possible that the judges have made a mistake and given you a list that could not correspond to any tournament. This would not be the first time the judges have made a mistake.

Please write a program to determine if the list of matches corresponds to a valid tournament.

Input

The first line of input contains N ($1 \leq N \leq 17$), which is the number of rounds in the tournament.

The next $2^N - 1$ lines describe the matches. Each of these lines contains two integers a ($1 \leq a \leq 2^N$) and b ($1 \leq b \leq 2^N$ and $a \neq b$), which are the two players that were paired off in this match.

Output

Display `OK` if the list of matches corresponds to a valid tournament, otherwise, display `MISTAKE`.

Sample Input 1

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

Sample Output 1

```
OK
```

Sample Input 2

```
3
1 2
3 5
5 3
1 7
2 4
3 2
6 8
```

Sample Output 2

```
MISTAKE
```

This page is intentionally left (almost) blank.

Problem E
Electrical Grids
Time limit: 4 seconds

The International Corporation for Power Connection (ICPC) has been asked by several Pacific Islands to interconnect their power grids for better reliability. Each island has exactly one power grid comprised of one or more power stations. All power stations of a power grid are connected directly or indirectly by power lines.

To interconnect the power grids of two islands, ICPC installs a new submarine power cable between two power stations, one on each island. The new cable comes at a cost that depends on which stations are interconnected. It also requires upgrades of the power grids on both interconnected islands. To upgrade the power grid on an island, ICPC upgrades a selection of its power lines. The chosen lines must connect all stations in the island's power grid. Upgrades caused by installing different submarine cables are independent of each other: the power grid on an island may have to be upgraded several times.

ICPC asks you to find the minimal total cost for interconnecting all power grids. The final grid must connect all power stations through power lines or submarine power cables.

Input

The first line of the input contains an integer n ($1 \leq n \leq 500$), which is the number of power stations across all islands. The next $n-1$ lines describe the interconnection and upgrading costs.

The i^{th} line of the input contains $i-1$ integers c_{ij} ($-10^9 \leq c_{ij} \leq 10^9$ for $1 \leq j < i \leq n$). If $c_{ij} < 0$ then stations i and j are on the same island and connected by a power line, one upgrade of which costs $-c_{ij}$. If $c_{ij} = 0$ then stations i and j are not directly connected by a power line and cannot be connected by a submarine power cable. If $c_{ij} > 0$ then stations i and j are on different islands and can be connected by installing a new submarine power cable at cost c_{ij}.

Output

Display the minimal total cost for interconnecting all power grids. If they cannot be interconnected, display `impossible` instead.

Sample Input 1
```
6
100
-1 200
10 40 70
20 50 80 -2
30 60 90 -4 -3
```

Sample Output 1
```
61
```

Sample Input 2
```
4
0
-1 0
0 -1 0
```

Sample Output 2
```
impossible
```

This page is intentionally left (almost) blank.

Problem F
Find and Replace
Time limit: 4 seconds

Jenny is about to submit her answers to a multiple-choice exam. Each question has five possible options (`A`, `B`, `C`, `D`, `E`), exactly one of which is correct. Jenny must select exactly one choice for each question that she thinks is the correct answer and she gets one point per problem that she answers correctly.

In the last minute of the exam, Jenny caught a glimpse of the answer key (that is, she saw the correct answer for each question). She does not have enough time to change all of her answers to the correct answers, but she can make use of the 'Find and Replace' feature. In a few clicks, she can change all of one answer to any possible answer. For example, she can change every `A` response that she has made to a `C` response.

Given that she only has enough time to do at most one Find and Replace, what is the maximum number of points that Jenny can achieve?

Input

The first line of input contains exactly one integer n ($1 \leq n \leq 200\,000$), which is the number of questions on the exam.

The second line describes Jenny's answers to the questions on the exam. The third line describes the correct answers. Each of these lines contains a string of length n consisting only of the characters `A`, `B`, `C`, `D`, and `E`.

Output

Display the maximum number of points that Jenny can achieve.

Sample Input 1

```
4
ABCD
ABDE
```

Sample Output 1

```
3
```

Sample Input 2

```
1
A
A
```

Sample Output 2

```
1
```

This page is intentionally left (almost) blank.

Problem G
Get Rect, Mate!
Time limit: 10 seconds

The world is in danger and the only way our hero Max can save it is by playing a children's card game. In this game, Max's opponent places several red axis-aligned rectangles on the floor. These rectangles may or may not overlap. Then, the opponent will place a blue axis-aligned rectangle, which may or may not overlap other rectangles. It is now up to Max to move the blue rectangle so that it is inside of the *slide area* (an axis-aligned rectangle) and does not overlap with any other rectangle. If Max can successfully do this, he is awarded a *good slider*. Note that the outside of the blue rectangle may touch the edge of a red rectangle or the edge of the slide area, but the inside of the blue rectangle must not overlap any red rectangle and must be inside the slide area. Originally, the blue and red rectangles may or may not be in the slide area.

If Max is awarded a good slider that moves the blue rectangle the minimum distance possible, then he gets *Rect*–a special bonus in the game, which Max greatly desires. While moving the blue rectangle, it may only be translated, not rotated or flipped.

Given the location of the rectangles and the slide area, please help Max get Rect by determining the minimum distance that the blue rectangle needs to move so that Max can be awarded a good slider.

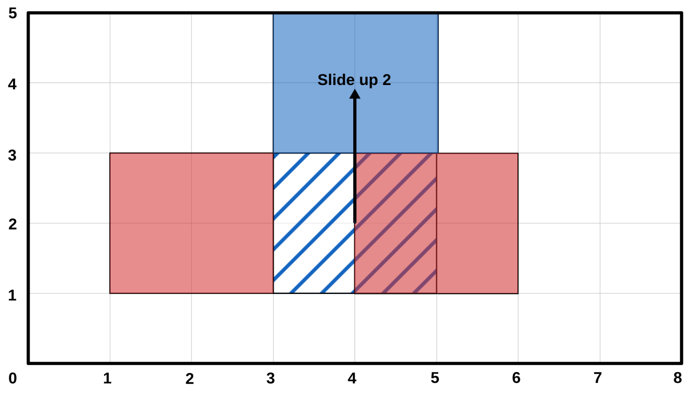


Illustration of Sample Input 1

Input

The first line of the input contains a single integer N ($1 \leq N \leq 5\,000$), which denotes the number of red rectangles.

The second line contains four integers S_x, S_y ($-10^6 \leq S_x, S_y \leq 10^6$), which denote the x- and y-coordinate of the bottom-left corner of the slide area, and S_w and S_h ($1 \leq S_w, S_h \leq 10^6$), which denote the width and height of the slide area.

The third line contains four integers B_x, B_y ($-10^6 \leq B_x, B_y \leq 10^6$), which denote the x- and y-coordinate of the bottom-left corner of the blue rectangle, and B_w and B_h ($1 \leq B_w, B_h \leq 10^6$), which denote the width and height of the blue rectangle.

The next N lines describe the red rectangles. Each of these lines contains four integers R_x, R_y ($-10^6 \leq R_x, R_y \leq 10^6$), which denote the x- and y-coordinate of the bottom-left corner of this rectangle, and R_w and R_h ($1 \leq R_w, R_h \leq 10^6$), which denote the width and height of this rectangle.

Note that x increases as you move right and y increases as you move up.

Output

Display the square of the minimum distance for getting Rect. If it is impossible to get Rect, display -1.

Sample Input 1

```
2
0 0 8 5
3 1 2 2
1 1 2 2
4 1 2 2
```

Sample Output 1

```
4
```

Sample Input 2

```
2
0 0 8 5
3 1 2 2
1 1 2 2
2 1 2 2
```

Sample Output 2

```
1
```

Sample Input 3

```
1
1 1 1000 1000
2 2 50 50
500 500 5 5
```

Sample Output 3

```
0
```

Sample Input 4

```
6
0 0 8 5
3 1 2 2
1 1 2 2
4 1 2 2
7 1 2 2
1 4 2 2
4 4 2 2
7 4 2 2
```

Sample Output 4

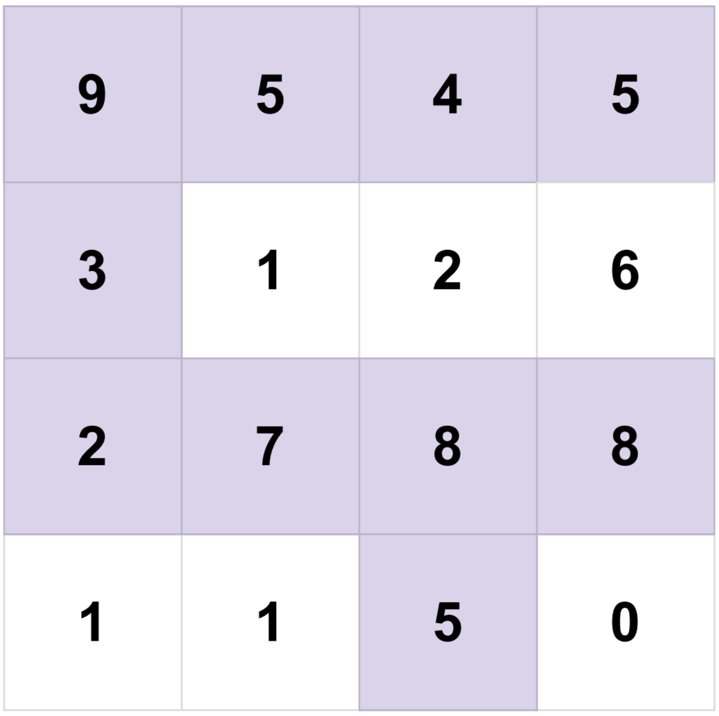
```
-1
```

Problem H
Housing
Time limit: 2 seconds

Jane has a plot of land represented by a grid of unit squares. She wants to build a house on this plot of land. Each square of the grid has an associated value which represents how good it is to build on that square. Jane also has a house plan. This plan defines the shape of the house Jane wants to build. Jane wants to place the house in a way that maximises the sum of grid values that it covers.

The house plan is a polyomino. A polyomino is a contiguous shape formed by one or more unit squares that are joined edge to edge. One way to define a polyomino is to start at some square, then walk either left, right, up, or down repeatedly. This walk may visit a square more than once, however, any such square should only be counted once when computing the value of a house placement. The squares visited at least once during this walk form a polyomino.

In the example below, the purple squares denote the polyomino formed by starting in the top-right square, then walking left three times, then down two times, then right twice, then down once, then up once, and then right once. The value for this placement of Jane's house is 56:



Jane can place her polyomino house plan anywhere on the grid. It is also allowed for some or all of her house to go off the grid, as Jane also owns land around her plot. However, as the surrounding land is barren wasteland, any squares off the grid should be considered to have zero value. Her house plan cannot be rotated. This would ruin Jane's view of the sunset.

Can you help Jane find the highest-value location to build her house?

Input

The first line of input contains M $(0 \leq M \leq 10)$, which is the length of the walk defining the polyomino house plan. The next line contains M letters, each of which is either `u` (representing up), `d` (representing down), `l` (representing left), or `r` (representing right). These letters define the polyomino walk.

The next line contains two integers R $(1 \leq R \leq 50)$ and C $(1 \leq C \leq 50)$, which are the number of rows and columns in Jane's plot of land respectively. Following are R lines each containing C integers. Each of these lines describes the values in the corresponding row of the grid. Each such value has an absolute value of no more than 1000.

Output

Display the maximum possible value some placement of Jane's house can achieve.

Sample Input 1

```
3
llu
4 4
1 2 -3 2
4 -2 4 6
8 -3 -3 -3
1 1 1 1
```

Sample Output 1

```
11
```

Sample Input 2

```
0

2 1
1
2
```

Sample Output 2

```
2
```

Sample Input 3

```
10
uuuullllldd
2 2
-99 -99
99 -99
```

Sample Output 3

```
99
```

Problem I
Item(s)
Time limit: 1 second

Have you noticed that an automated check out system sometimes asks you to take your "items" (plural) when you buy only one item?

In this problem, you are asked to tell the difference between one and more than one in a check out style situation, in which someone first scans some items to purchase, then might try to remove some items (for example, after seeing the total cost). There is a list of item types that have been scanned and the count of each type. Each type occurs only once in this list. There is also a (possibly empty) list of item types that are then removed and the count of each type that is removed. Again, each type occurs at most once in this list.

When some item type occurs with a count greater than 1 in a list, the type will be shown in plural form. The plural form of a type consists of the singular version of the item type with an "s" added. For example, the plural of mouse is "mouses", and the plural of sheep is "sheeps".

You must process the two lists. There is an *error* if some item has been removed more than its count in the list of scanned items. It is a *null* transaction if all the items in the list of scanned items are removed. There is an *item* if exactly one item remains, otherwise, there are *items*.

Input

The first line of input contains two integers S $(1 \leq S \leq 100)$, which is the number of scanned item types, and R $(0 \leq R \leq 100)$, which is the number of item types to be removed.

The next S lines contain the list of scanned items. Each line contains an integer C $(1 \leq C \leq 100)$, which is the count of items of this type scanned, and then the name of the item type in singular or plural form as appropriate to the item count. The singular form of an item type name consists of L $(1 \leq L \leq 100)$ lowercase letters. The next R lines contain the list of items to be removed in the same format as the list of scanned items.

Output

Display either `error`, `null`, `item`, or `items` as appropriate.

Sample Input 1	Sample Output 1
1 1 2 sheeps 1 dog	error

Sample Input 2	Sample Output 2
1 1 2 mouses 2 mouses	null

Sample Input 3	Sample Output 3
1 1 2 sheeps 1 sheep	item

Sample Input 4	Sample Output 4
1 0 2 sheeps	items

Sample Input 5

```
1 1
2 grasss
1 grass
```

Sample Output 5

```
item
```

Problem J
Junipers
Time limit: 10 seconds

Jack and Jill have been busy planning their juniper garden. They have cleared out an area in the backyard and have divided the land into a $g \times g$ grid. Any number of junipers can be planted in each of the g^2 plots.

Jack and Jill have already purchased the junipers they wish to plant. Certain junipers have constraints on where they can be located in the garden relative to other junipers. Jack and Jill have written down their constraints in two forms: row-constraints and column-constraints. A row-constraint between two junipers specifies that juniper x must be planted between s and t rows (inclusive) further south than where juniper y is planted (but not necessarily in the same column). A column-constraint between two junipers specifies that juniper x must be planted between s and t columns (inclusive) further east than where juniper y is planted (but not necessarily in the same row).

Help Jack and Jill plant the junipers in their garden.

Input

The first line of input contains four integers n ($1 \leq n \leq 250$), which is the number of junipers, g ($1 \leq g \leq 1\,000\,000\,000$), which is the size of one side of the garden, R ($0 \leq R \leq 62\,500$), which is the number of row-constraints, and C ($0 \leq C \leq 62\,500$), which is the number of column-constraints.

The next R lines describe the row-constraints. Each of these lines contains four integers x ($1 \leq x \leq n$), y ($1 \leq y \leq n$ and $x \neq y$), s ($0 \leq s < g$), t ($s \leq t < g$), signifying that the row in which tree x is planted must be at least s and no more than t larger than the row in which tree y is planted.

The next C lines describe the column-constraints. Each of these lines contains four integers x ($1 \leq x \leq n$), y ($1 \leq y \leq n$ and $x \neq y$), s ($0 \leq s < g$), t ($s \leq t < g$), signifying that the column in which tree x is planted must be at least s and no more than t larger than the column in which tree y is planted.

The trees are numbered $1, \ldots, n$. Rows are numbered $1, \ldots, g$ from north to south. Columns are numbered $1, \ldots, g$ from west to east.

Output

If there is a valid arrangement of junipers such that all the constraints are met, display the row and column number of the junipers in the order $1, 2, \ldots, n$. If there are multiple valid arrangements, any one will be accepted.

If there is no valid arrangement of junipers, display -1 instead.

Sample Input 1

```
3 5 3 2
1 3 1 2
1 2 2 3
3 2 1 1
1 2 2 3
2 3 2 3
```

Sample Output 1

```
5 5
3 3
4 1
```

Sample Input 2

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

Sample Output 2

```
-1
```

This page is intentionally left (almost) blank.

Problem K
Karry Katastrophe
Time limit: 9 seconds

Delaney has just finished creating his very first adding machine! Given two integers (in base 10), it will output the sum of the two numbers—at least, it was supposed to! Delaney has made a mistake when making his machine and forgot to implement the carry operation. This means that each column of digits is added together independently. If the sum of a column is at least 10, then the machine will only care about the least significant digit.

For example, $31415 + 9265 = 30670$.

Since it will take Delaney months to implement the carry operation, he has decided to play a game with his friends instead: given a list of numbers, he wants to find two distinct numbers from this list that will give the largest 'sum' according to Delaney's machine. Note that if a number appears in the list two or more times, they may be summed together. What is the largest 'sum'?

$$\begin{array}{rr} & 1 \\ & 27 \\ + & 59 \\ \hline & 86 \end{array}$$

Input

The first line of input contains a single integer n ($2 \leq n \leq 200\,000$), which is the number of integers in the list.

The next n lines describe the list. Each of these lines contains a single integer x ($0 \leq x \leq 10^{18}$), which is an element in the list.

Output

Display the largest 'sum' of two distinct numbers in the list.

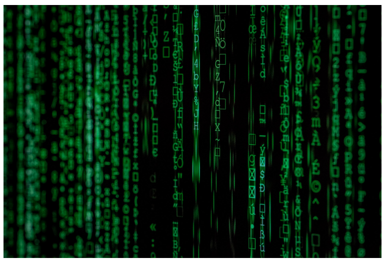
Sample Input 1
```
2
31415
9265
```
Sample Output 1
```
30670
```

Sample Input 2
```
3
123
456
789
```
Sample Output 2
```
802
```

Sample Input 3
```
2
1234
9875
```
Sample Output 3
```
9
```

Sample Input 4
```
3
123
123
456
```
Sample Output 4
```
579
```

Problem L
Lost Matrix
Time limit: 1 second



It has been a year of non-stop planning and work, but you have finally done it! You finished your masterpiece of modern art. Specifically, you created a base for your art piece which is a 2-dimensional grid with each of the $n \times n$ cells being capable of holding any number of abstract shapes.

You modelled your art piece after a 2-dimensional $n \times n$ matrix of non-negative integers. For every number in the matrix, you put that many abstract shapes in the corresponding cell in your art piece.

After stepping back, and admiring your creation from afar, you observed just how perfect it is. But, a few visitors to the museum housing your art bumped into your art piece and knocked it over! Nearly everything fell off and your creation was left in ruins.

You want to reassemble things, but you cannot quite remember exactly how many abstract shapes were in each corresponding cell. However, you know the total number of abstract shapes that were in each row and in each column. So, you decided to write a computer program to use this information to reassemble your art piece.

Input

The first line of input consists of a single integer n ($1 \le n \le 100$), which is the number of rows and columns in the original matrix after which your piece is modelled.

The second line contains the n row sums of the matrix. These row sums are given in order from top row to bottom row and each sum is in the inclusive range from 0 to 1 000.

The third line contains the n column sums of the matrix. These column sums are given in order from leftmost column to rightmost column and each sum is in the inclusive range from 0 to 1 000.

Output

Display an $n \times n$ matrix that has the appropriate row and column sums. All values in the matrix must be non-negative integers. Any valid solution will be accepted.

If no valid solution exists, display -1.

Sample Input 1

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

Sample Output 1

```
0 0 0 1
0 0 2 0
0 3 0 0
4 0 0 0
```

Sample Input 2

```
3
1 5 4
2 3 4
```

Sample Output 2

```
-1
```

This page is intentionally left (almost) blank.

Problem M
Marvelous Marble Machine
Time limit: 1 second

Mary and Marty have finished work on their Marvelous Marble Machine. The machine takes in a mountain of marbles, mixes them up, and then ejects the marbles out one at a time. Mary and Marty's marbles come in three types: single stripe, double stripe, and triple stripe.

Mary and Marty have been testing their Marvelous Marble Machine and they want to know if the machine is working properly. The machine is working properly as long as the marbles put in are the same as the ones the machine ejected. This means that the counts of each type are the same. Is Mary and Marty's Marvelous Marble Machine working properly?

Input

The first line of input contains a single integer n ($1 \leq n \leq 100$), which is the number of marbles Mary and Marty put into their machine.

The second line contains n integers describing the marbles that were put into the machine. Each integer is either 1, 2, or 3 representing the number of stripes on the marble.

The third line contains n integers describing the marbles that the machine ejected. Each integer is either 1, 2, or 3 representing the number of stripes on the marble.

Output

Display `marvelous` if the machine is working properly. Otherwise, display `error`.

Sample Input 1	Sample Output 1

5
1 2 3 1 3
2 3 1 1 3
	marvelous

Sample Input 2	Sample Output 2

5
1 2 3 1 3
2 3 1 2 3
	error

This page is intentionally left (almost) blank.