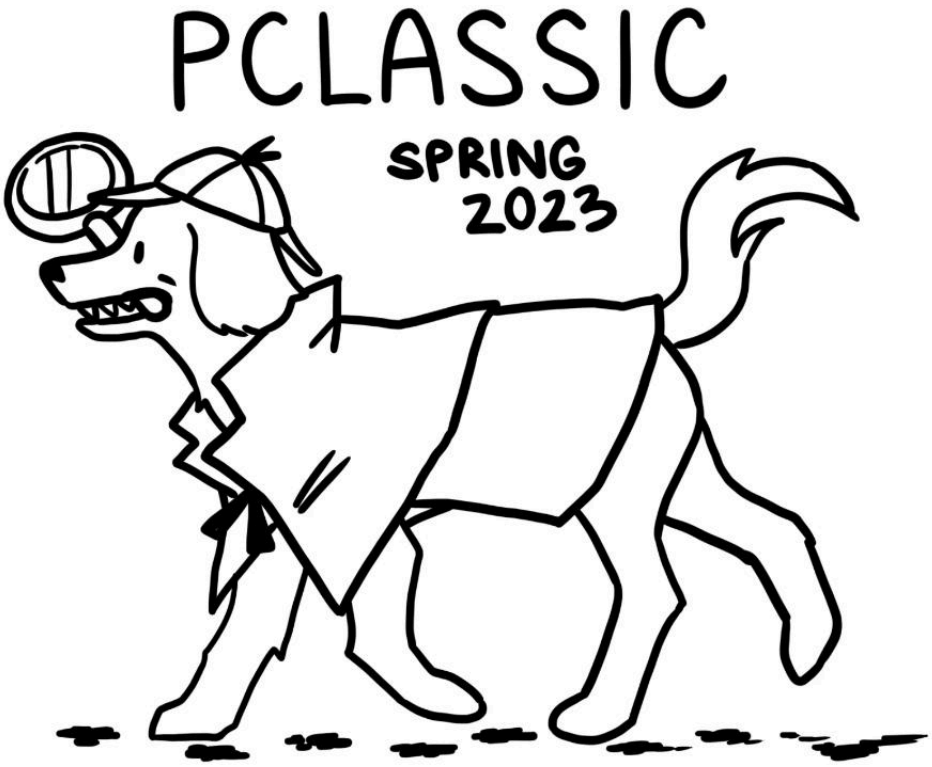
Contest Problems

Philadelphia Classic, Spring 2023

University of Pennsylvania



Rules and Information

This document includes 12 problems. **Classic teams do problems 1-8; Advanced teams do problems 5-12**.

Any team which submits a correct solution for any of problems 1-4 will be assumed to be a classic team. **If you are not a classic team, please skip problems 1-4.**

Problems 1-4 are easier than problems 5-8, which are easier than problems 9-12.

You may use the Internet only for submitting your solutions, reading Javadocs or Python documentation, and referring to any documents we link you to. You **may not** use the Internet for things like StackOverflow, Google, AI assistance or outside communication. Please do not use ChatGPT.

You are responsible for handling Input and Output on your own.

You will receive 1 point per correct submission. In the case of a tie, the 'total penalty' will be used as a tiebreaker, which is defined to be the sum of time elapsed from the beginning of the contest to the first accepted submission for each problem, plus a 10 minute penalty for each previously rejected run for that problem. Wrong Answer, Time Limit Exceeded, and Runtime Errors all contribute towards the penalty. Unsolved problems do not contribute towards the penalty. There is no partial credit.

Some problems use Java's "long" type; if you are unfamiliar with them, they're like an "int", but with a (much) bigger upper bound, and you have to add "L" to the end of an explicit value assignment:

```
long myLong = 1000000000000L;
```

Otherwise, the "long" type functions just like the "int" type.

INTRODUCTION

Believe it or not, puppies and kittens have been waging a hairy war for centuries. I'm Agent Ishaan, the head of the K-9 unit, and we received horrifying news this morning!

In front of you lies the body of Ethan Chee, a renowned foster puppy rescuer! We found his body this morning brutally snuggled to death. Our experts think that this is a MURDER done by the notorious kittens in the Evil International Kitten Society! Even worse, all of Ethan's puppies are missing...

The only thing we found in his pockets is his phone which we believe has the information that can lead us to his murderer and the missing puppies. Using your CS skills, you need to solve as many of the clues as possible to find the kitten that murdered him and rescue the kidnapped puppies!

1. Puzzling Password

The first thing we need to do is to unlock Ethan's phone, which is locked by a simple lock. The lock takes in two integers **n** and **k** and returns the remainder of when n is divided by k. Help us unlock his phone by coding the algorithm for the lock.

Constraints:

$1 \leq n \leq 10^9$

$1 \leq k \leq 10^9$

Input:

First line contains **n k** separated by a space

Output:

For each test case return **the remainder when n is divided by k.**

Sample Test Cases

Sample Input	Sample Output	Explanation
16 2	0	The remainder when 16 is divided by 2 is 0
100 3	1	The remainder when 100 is divided by 3 is 1
52 30	22	The remainder when 52 is divided by 30 is 22.

2. Counting Canines

We need to get an estimate on the number of puppies that were kidnapped. In his Notes app, Ethan wrote four integers: **a, b, c, d**. The number of puppies he took care of is the maximum absolute value of the sum of any two of these integers.

Constraints:

$-10^9 \leq a, b, c, d \leq 10^9$

$T \leq 10^4$

Input:

The first line contains the number of test cases **T**.
The next T lines contain the 4 integers **a b c d** separated by a space.

Output:

Return the maximum absolute value of the sum of two

Sample Test Cases

Sample Input	Sample Output	Explanation
1 1 2 3 4	7	|1+2|=3, |1+3|=4, |2+3|=5, and |3+4|=7. The maximum is 7.
1 -10 -3 2 2	13	|-10+(-3)|=13 which is the maximum.

2 1 1 1 1 −1 −2 −3 −4	2 7	|1+1|=2 which is the maximum of the first test case. |-3+(-4)|=7 which is the maximum of the second test case.

3. Finding Fido

Wow, that's a whole lot of puppies! We've received information that the puppies are probably still with the murderer! We've gotten access to his FindMyIphone map, which comes in the form of an **n x n 2D array of integers.**
The map is a 2D array with n rows and n columns. The array contains distinct numbers 1, 2, 3 ... to n^2 except one of the numbers is missing. The missing number has been replaced with a -1. We think that the missing number in the array is the location of the murderer and the puppies! Help determine what the missing number is.

Constraints:

$1 \leq n \leq 500$

Input:

The first line contains **n**
The **i**th of the next **n** lines contains **n** integers $a_{i1}, a_{i2}, \ldots, a_{in}$ **(-1 <= a_{ij} <= n^2, a_{ij} != 0)**

Output:

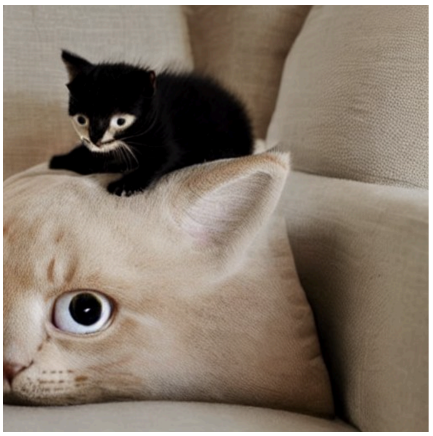
Return the missing number

Sample Test Cases

Sample Input	Sample Output	Explanation
3 7 2 4 -1 5 6 3 9 8	1	The missing number is 1
4 15 1 11 13 2 -1 7 8 9 10 16 4 5 6 12 14	3	The missing number is 3

1 -1	1	The missing number is 1

4. The International Kitten Sofa



It looks like the kittens are located in a mysterious complex called the International Kitten Sofa. We've hacked into their communication channels, and we've intercepted a cryptic text message. We know that the kidnapped puppies are being moved around, and we believe that the message will reveal where the puppies are being transported. It seems like the International Kitten Kidnapping Society sends encrypted messages using an encryption method.

The text message we've received is a string of English letters **s1**. We've been tracking the Society for a while, so we have the decryption key in the form of a binary string **s2.** Recall that a binary string is a string whose characters are 0 or 1. In **s2,** the character 1 represents a vowel and the character 0 represents a consonant. Help decrypt the message by counting the number of times the vowel-consonant pattern represented by **s2** appears in **s1**!

Constraints:

$1 \leq |s2| \leq |s1| \leq 2000$

Input:

The first line contains **s1**
The second line contains **s2**

Output:

Output a single integer – the number of times the vowel-consonant pattern represented by **s2** appears in **s1**

Sample Test Cases

Sample Input	Sample Output	Explanation
banana 101	2	b**ana**na and ban**ana** provide 2.
Pclassic 100	1	Pcl**ass**ic is the only 1
AaaaBBbeee 11	5	**Aa**aaBBbeee, A**aa**aBBbeee, Aa**aa**BBbeee, AaaaBBb**ee**e, AaaaBBbe**ee**

SUMMARY

Our K-9 unit has done some preliminary work on the case. They've already narrowed the location of the puppies and the killer kittens down to a mysterious complex called the International Kitten Sofa. They've also decrypted a message that tells us the kidnapped puppies are in transit to the Sofa.

5. Interesting Iditarod

Based on the text message, the kidnapped puppies will be transferred to the International Kitten Sofa very soon. Let's use our trusty dog sled to get there as fast as possible!
The dog sled is very particular and is only able to travel in step sizes which are powers of **a** dog-steps long per minute. Recall the powers of a are a^0, a^1, a^2, a^3, etc**.** We need to travel **b** dog-steps in total. Help us calculate the fewest number of minutes it will take to reach the murderer!

Constraints:

$1 <= T <= 10^4$

$1 <= a, b <= 10^9$

Input:

The first line contains one integer **t** - the number of test cases
The first and only line of each test case contains two space separated integers representing **a** and **b**

Output:

An integer representing the minimum number of steps it takes to reach finish position **b.**

Sample Test Cases

Sample Input	Sample Output	Explanation
1 1 100	100	Can only make steps of length 1
1 2 6	2	1 step of length 4 to get to 4 1 step of length 2 to get to 6

2 1 100 2 6	100 2	Same as the first two, but notice the first line is 2 denoting there are 2 test cases.
1 10 925	16	9 steps of length 100 to get to 900 2 steps of length 10 to get to 920 5 steps of length 1 to get to 925

6. Cat Chase Dog-sled

We need to break into the International Kitten Sofa. Unfortunately, there is a mean-looking Siamese kitten on the prowl. Luckily, our canines can sniff out exactly where the kitten will go.

We are on a number line from x = 0 to x = n - 1 that wraps around (moving right at position x= n-1 goes to position x = 0, and moving left at position x = 0 goes to position x = n-1). The Siamese kitten starts at x = c and we start at x = r. The Siamese kitten can make a move 1 unit in either direction every second. Each second, our dog sled can also move 1 unit in either direction or stay in the same spot. If the Siamese kitten and our dog sled are ever in the same position, then the dog sled has been caught. Given a string of L and R (left/right) moves of length p, which is the Siamese kitten's move at each time step, what is the minimum number of moves that we can make that still ensures we are not caught by the evil kitten?

Constraints:

$2 \leq n \leq 10^9$

$0 \leq c < n$

$0 \leq r < n$

$c \neq r$

$2 \leq p \leq 2 * 10^5$

Input:

The first line contains **n c r p**, which is the size of the number line, the initial location of the Siamese kitten, the initial location of our dog sled, and the size of the kitten's path respectively.

The next line contains p characters of L or R representing the kitten's path.

Output:

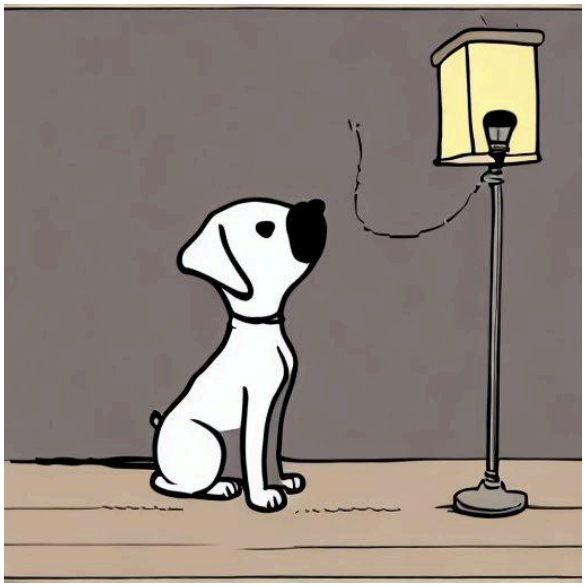
For each test case, return an integer representing the minimum number of moves the dog sled has to move without getting caught.

Sample Test Cases

Sample Input	Sample Output	Explanation
6 0 1 3 RRR	3	This is the initial state: CR.... After the 1st turn (**R**RR): .CR... After the 2nd turn (R**R**R): ..CR.. After the 3rd turn (RR**R**): ...CR.
10 2 6 3 LRL	0	Initial state: ..C...R... After the 1st turn (**L**RL): .C....R... After the 2nd turn (L**R**L): ..C...R... After the 3rd turn (LR**L**): .C....R...

6 2 0 2 LL	1	Initial state: R.C... After the 1st turn (**L**L): RC.... After the 2nd turn (L**L**): C....R

7. Bright-Eyed and Bushy-Tailed



We've gotten past the Siamese kitten guard and are now in the main room. The International Kitten Sofa is adapted for cats, and not humans, so the entire room is dark, making it difficult to find the puppies. Luckily, there are lightbulbs arranged in a line.

There are **n** lightbulbs, each placed on an integer position on a number line. Each lightbulb has a brightness value, which affects that location and the surrounding location. There is a linear decay in brightness, so a lightbulb at position **i** with brightness **j** will give j brightness to position i, j-1 brightness to position i-1 and i+1, etc., until 0 brightness at position i-j and i+j and beyond. You may assume only integer positions on the line are affected by lightbulb brightnesses. Multiple lightbulbs can be placed at the same position, and the brightness at a position is the sum of the brightness of all lightbulbs. Your goal is to return an integer representing the brightness value of the brightest point. Help us find the brightest spot so that we can see the evil kittens in the Sofa!

Constraints:

1 <= n <= 2000

-10^5 <= positions <= 10^5

0 <= brightness <= 10^5

Input:

The first line contains **n**, the number of lightbulbs
The second line contains **n** space-separated integer positions of the lightbulbs.
The third line **n** space-separated integer brightnesses of the lightbulbs.

Output:

Return an integer representing the brightness value of the brightest point.

Sample Test Cases

Sample Input	Sample Output	Explanation
2 -1 1 2 2	2	The max brightness can be found at points -1, 0, and 1, all of which have brightness values of 2.
3 1 3 4 3 4 2	6	The max brightness is at point 3, which has a brightness value of 6.

8. EQUAL FOOTING

Great job! You've found the brightest spot in the room, and as your eyes adjust, you can see the room is littered with stacks of cardboard boxes. Across the room, a pair of amber eyes emerge, staring at you and your band of puppies. Out of the shadows, a kitten saunters into the limelight. But wait! Is this even a kitten? The kitten has orange and black fur, sharp teeth and claws, and - no, it *can't* be - it's a tiger kitten!
Suddenly, the room transforms into a narrow hallway! You can only move in a straight line now. The height of the tiles also changes. Luckily, you can always reach the next or previous tile because the height value of adjacent locations differ by exactly 1. The height of a tile is always greater than or equal to 0. The tiger kitten starts on one end of the hallway, and you start on the other end. The tiger kitten wants to see if you are worthy of its friendship, so it decides to play a game with you.

At each time step, you and the kitten **must** move one step, either forward or backward. Furthermore, at each time step, you and the kitten must be in positions of the **same height** value. Both of you want to reach the other side of the hallway at the same time. What is the (integer) minimum number of time steps it takes for both you and the kitten to reach your respective goals?

Note that if one of you reaches the other side of the hallway first, you are not required to stay there. The only way to earn the tiger kitten's trust is to have both you and the tiger kitten reach their goals at the exact same time.

Constraints:

$1 <= n <= 2000$

$|heights[i] - heights[i+1]| = 1$

$1 <= heights[i] <= 10^9$

$heights[1] = heights[n] = 1$

Input:

The first line contains the length **n** of the hallway.
The second line contains n space-separated integer heights for each position of the hallway from where you start to where the tiger kitten starts. We guarantee that the first height and last height are both 1.

Output:

Return an integer representing the minimum number of time steps it takes for both you and the tiger kitten to reach your respective goals.

Sample Test Cases

Sample Input	Sample Output	Explanation
3 1 2 1	2	Left: 1 → 2 → 3 Right: 3 → 2 → 1 It takes 2 steps.
7 1 2 1 2 3 2 1	8	Left: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 6 → 7 Right: 7 → 6 → 7 → 6 → 5 → 4 → 3 → 2 → 1 It takes 8 steps.
13 1 2 3 2 3 4 3 2 1 2 3 2 1	18	Left: 1 → 2 → 3 → 2 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 7 → 8 → 9 → 10 → 11 → 12 → 13 Right: 13 → 12 → 11 → 10 → 9 → 8 → 7 → 8 → 7 → 6 → 5 → 4 → 3 → 2 → 1 → 2 → 3 → 2 → 1 It takes 18 steps.

9. You Are A Path-cifist

You've befriended the almighty tiger kitten who spits out a furball containing a map to the puppies. The map is a 2D array filled with the number of evil kittens in each cell as well as the source cell and the target cell. You want to find a path from the source to the target cell such that you are passing and fighting the minimum number of evil kittens. The source cell is given in terms of starting row and column (srcr and srcc), and the target cell is given in terms of ending row and column (endr and endc). You are only able to move up, down, left, or right (if such a movement is allowed). You cannot move diagonally. Each coordinate contains the number of evil kittens which you will have to fight if you pass through the coordinate.

Constraints:

$1 \leq n, m \leq 1000$

$1 \leq srcr, endr \leq n$

$1 \leq srcc, endc \leq m$

$0 \leq a_{ij} \leq 10^7$

Input:

For each test case:
The first line contains two integers **n**, **m**, the number of rows and columns of the 2D array.
The **i**-th of the next **n** lines contains **m** integers $\mathbf{a_{i1}}$, $\mathbf{a_{i2}}$, …, $\mathbf{a_{im}}$ – the number of evil kittens in the cell at the **i**th row and **j**th column
The next contains **srcr srcc**, which are the row and column of the starting cell.
The next line contains **endr endc**, which are the row and column of the ending cell.

Output:

For each test case, return an integer representing the lowest number of obstacles to remove traveling from (srcr, srcc) to (endr, endc)

moving only vertically or horizontally. You are guaranteed that every input has a solution.

Sample Test Cases

Sample Input	Sample Output	Explanation
1 5 1 2 3 4 5 1 1 1 4	10	We start at a_{11} = 1 and traverse through a_{41}. 1 + 2 + 3 + 4 + 5 = 10
4 2 0 4 5 0 3 3 2 2 1 1 4 2	9	We start at a_{11} = 0 and traverse through a_{24} = 2. RIGHT DOWN DOWN DOWN 0 + 4 + 0 + 3 + 2 = 9
5 5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 3 3 5 4	70	We start at a_{33} = 13 and traverse through a_{45} =24 . RIGHT, DOWN, DOWN 13 + 14 + 19 + 24 = 70
6 3 4 10 12 3 4 5 1 4 6 4 4 5 5 0 0 7 8 10 5 2 1 1	16	We start at a_{25} = 0 and traverse through a_{11} = 4 . UP, LEFT, UP, UP, UP 0 + 4 + 4 + 1 + 3 + 4 = 16

10. I N EE DAP UPPYR ADARNOW!

You've reached the location of the puppies. The puppies are hidden behind a bunch of doors. Luckily, there's a vending machine that sells a coveted Puppy Radar. It looks like a metal detector, but it actually detects puppies! The vending machine has a weird payment system:

The Puppy Radar costs **n** cents. In order to pay n cents, you can only enter pennies (1 cent), nickels (5 cents), dimes (10 cents), quarters (25 cents), and dollar coins (100 cents). Furthermore, the vending machine keeps track of **k** which is the number of coins you use and will only return the Puppy Radar to you if **k** is a Fibonacci number. You want to use **k** coins to pay the **n** cents where **k** is as small as possible and also is a Fibonacci number. Return the number of coins **k** or -1 if it is not possible to buy the Puppy Radar.

Note: A fibonacci number is part of a series of numbers in which each number is the sum of the two preceding numbers, with the first two Fibonacci numbers being 1, 1. The first fibonacci numbers are: 1, 1, 2, 3, 5, 8...

Constraints:

$1 \leq n \leq 2 * 10^5$

$1 \leq t \leq 1000$

Input:

The first line represents the number of test cases **t**.
The first and only line of each test case contains a single integer **n** - the amount of money to be formed.

Output:

For each test case, return the minimum Fibonacci number of coins needed to form **n**, or **-1** if it is not possible.

Sample Test Cases

Sample Input	Sample Output	Explanation
5 1 2 3	1 2 3 -1	1¢ – 1 coin (1 penny). 2¢ – 2 coins (2 pennies). 3¢ – 3 coins (3 pennies). 4¢ cannot be made with a Fibonacci number of coins.

4 5	1	5¢ – 1 coin (1 nickel)
2 500 1000	5 13	500¢ – 5 coins (5 dollar coins). 1000¢ – 13 coins (9 dollar coins, 4 quarters).
4 106 155 170 187	3 5 5 8	106¢ – 3 coins (1 dollar coin, 1 nickel, 1 penny). 155¢ – 5 coins (1 dollar coin, 1 quarters, 3 dimes). 170¢ – 5 coins (1 dollar coin, 2 quarters, 2 dimes). 187¢ – 8 coins (1 dollar coin, 3 quarters, 2 dimes, 2 pennies).

11. Hidden Doggos

The puppies are hidden behind **n** doors, and each door has **a_i** puppies hidden behind the door. In order to rescue all the puppies in time, you need to know exactly how many puppies are behind each door. Luckily, the Puppy Radar you got from the vending machine can help you detect the puppies!
The Puppy Radar comes with the following instructions:
To make a query, output "? i j" ($1 \le i, j \le n, i \ne j$) without quotes. Afterwards, the Puppy Radar will produce an answer, and you should read the answer in as one integer, the value of $(a_i \oplus a_j) + a_j$.

Once you are finished with using the Puppy Radar, and you are confident that you know how many puppies are behind each door, you can tell your Puppy Extraction Team over the radio to extract the puppies.

When you are ready to give the final answer, output "! a_1 a_2 … a_n" without quotes, the calculated values of a. This does not count towards the number of queries. Make sure to output end of line and flush after printing a query

The Puppy Radar has a limited number of uses before it breaks down, so you will not be allowed to ask more than **n** queries. If you receive the integer −1 instead of an answer, it means your program has made an invalid query, has exceeded the number of queries in a test case, or has given an incorrect answer on the previous test case. Your program must terminate immediately to receive a Wrong Answer verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

Constraints:

$1 \le t \le 10^4$

$2 \le n \le 10^4$

$2 \le \sum n \le 10^4$

$0 \le a_i \le 10^9$

Input:

The first line represents the number of test cases **t**.
For each test case:
The first line contains an integer **n**
After reading the first line of every input, begin the interaction

Interaction:

To make a query, output "? i j" ($1 \le i,j \le n, i \ne j$) without quotes.
Afterwards, you should read in one integer, the value of $(a_i \oplus a_j) + a_j$.
If you receive the integer -1 instead of an answer, it means your program has made an invalid query, has exceeded the limit of **n** queries in a test case, or has given incorrect answer on the previous test case. Your program must terminate immediately to receive a Wrong Answer verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.
When you are ready to give the final answer, output "! a_1 a_2 … a_n" without quotes, the calculated values of a. This does not count towards the number of queries.
Make sure to output end of line and flush after printing a query.
PLEASE REFER TO THE CODEFORCES PROBLEM FOR MORE INFORMATION REGARDING INTERACTION.

Sample Test Cases

Sample Input	Sample Output	Explanation
1 3 ? 1 2 ? 2 3 ! 3 1 4	3 9	The hidden array is a=[3,1,4]. After querying "? 1 2", the output is $(3 \oplus 1) + 1 = 2 + 1 = 3$. After querying "? 2 3", the output is $(1 \oplus 4) + 4 = 5 + 4 = 9$. After finding the answer, they output the found array.
2 2 ? 1 2 ? 2 1 ! 8 9 3 ? 1 2 ? 3 2 ! 7 2 9	10 9 5 13	

12. PEACE TREAT-Y

After successfully rescuing the puppies, the kittens want to negotiate a peace agreement to end a century-long war. They've sent over an encrypted message in the form of two strings **s** and **t**. However, only a subset of characters **S** are part of the secret message. Find a maximum size subset **S** of characters from **s** and **t** such that when we remove all characters not in **S** from **s** and **t**, **s** and **t** turn into the same string. Return the characters in **S**

Constraints:

$1 \leq |s|, |t| \leq 10^5$

Input:

The first line contains a string **s**
The second line contains a string **t**

Output:

In the first line, output one integer **n** (0 <= **n** <= 26) - the number of elements in S.
In the second line, output a string - the set of characters, with each character in S appearing exactly once in the string.

Sample Test Cases

Sample Input	Sample Output	Explanation
abacb aabcb	2 ac	The largest subsequence both in s and t is "aac". This subsequence can be constructed using the letters from the subset {a, c}, which is ac when concatenated.

abc def	0	There is no common substring with respect to both strings except the empty string. As the empty string does not require any characters, we return an empty set concatenated to an empty string.
aacd dcaamb	1 a	The longest common substring to both characters is aa, which can be generated by the subset {a}.