PROSOLVE

8.0

TALK IS CHEAP, SHOWS US THE CODE

Competitive Programming

This problem set contains 10 questions (A-J)

23 November 2019

Organized by

Artificial Intelligence Society (AIS) UiTM Shah Alam

A. Hello, World?

Author: Syafiq

“Hello, World!” is commonly used example to express how does the language work, the syntax behind it and also demonstrate the I/O capability that build in the languages itself. How about we write a code to print the “Hello, World!” string?

Oh…. there is a catch, everytime you encounter string “Hello”, reverse it and put exclamation mark on the next word. Easy right?

Input

First line will be an integer ‘*n*’, which represent the number of cases, followed by ‘*n*’ number of strings. Length of the string will be *5 <= length(s) <=100.*

Output

The ‘*n*’ string of transformed string, as shown in the example input.

Sample Input/Output

Input	**Output**
3 Hello World hello World Morning everyone! Have you say Hello, World?	 olleH World! hello World Morning everyone! Have you say olleH, World!?

B. Wait? Why there is a plane infront me?!

Author: Syafiq

Do you know that there is a system that most of the plane flying in the air nowadays have some sort of traffic collision avoidance system? It is called as TCAS. It work by transmitting a radar signal around the plane itself where its range can conver upwards 40 miles radius. When other airplane detect the signal, the system will run its algorithm to avoid that collision and immediately notify the pilot whether to climb up or decent. One will receive climbing instruction, another would be decend instruction.

Your task is you need to build a similar system, but only in two 2D space and only telling there's a plane near us. No need to be complicated, just need to check whether there is a plane or not within the specified radius and alert if there is. If not, just proceed as usual.

Input

'n' number of plane, followed by 1 line of plane coordinate in Cartesian coordinate system that we need to ensure that the plane is not in-course toward a collision and last digit of the line is the radius of detection. After that, there is '*n*' number of line which specify the coordinate of other plane around it.

Output

Print a string "ALERT" if there is a plane around us, else do nothing

Sample Input / Output:

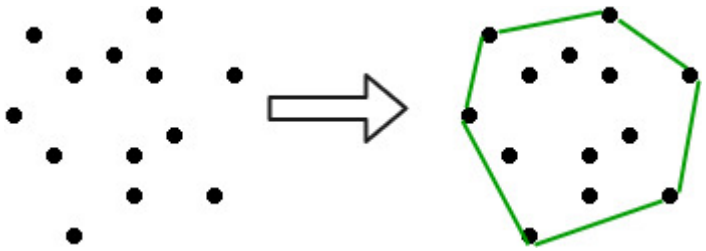
Input	Output
3 4 7 5 -4 0 10 4 -1 5	 ALERT

C. Encirclement

Author: Shahril

One of the most and ancient tactics in war is encirclement. It works by surrounding the position to isolate enemy forces. The most common reasons is to actually prevent outside reinforcements and supplies.

With having the enemy forces encircled, attacker can focus on attacking multiple enemy points in the inner circle. When this happens, defender (those inside the circle) either must fight to the death in order for them to break the encirclement, or surrender.



You, as *Admiral General Aladeen* most trusted programmer, has been instructed to create a program, which can calculate which enemy units to attack with the intention of creating encirclement for inner enemy units. One strict condition that he asked is to never let enemy units to be outside of the encirclement, or else disaster will happen. Beware, you don't want to disappoint Aladeen, right?

Input

First line of input is an integer ***N (3≤N≤100)***, where ***N*** is the number of enemy units.

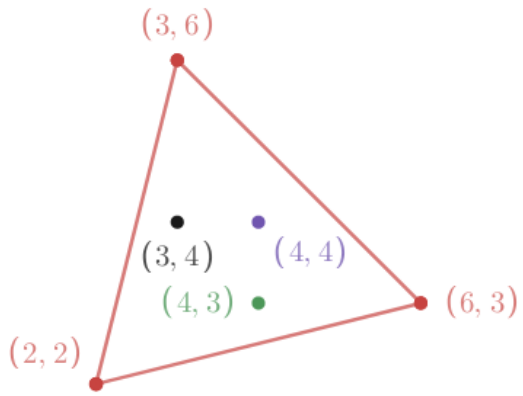
For the next ***N*** lines, there are two integers ***X*** and ***Y (0≤X,Y≤1,000)***, where these denote point coordinates of enemy units.

Output

Output all point coordinates of enemy units to attack based on *Admiral General Aladeen* instruction above. You need to sort the point coordinates result based on the formula ***(a.x+a.y) < (b.x+b.y)*** where ***a*** is the first point and ***b*** is the second point.

Sample Input	**Sample Output**
6 2 2 4 4 3 4 4 3 3 6 6 3	2 2 6 3 3 6

Explanation:



All point coordinates in sample output have created a close polygon, where inner enemy units have been totally circled.

D. Kakilator

Author: Syafiq

Ahem… I mean it is “calculator”, which I presume you all are familiar with it. Just implement a calculator according to specification below: -

Calculator Specs

- Only accept single digit number
- Result can be in two digit
- Accept math operator ‘+’, ‘-’, ‘*’, ‘/’
- Any other input, print out ‘INVALID’

Input

There is ‘*n*’ number of cases, which represent the number of input string to be fed into the calculator. Integer input will be in the provided range: *0 <= x <= 9.*

Output

The result of the computation, or string “INVALID” if input doesn’t met the constraint & specs.

Sample Input

Input	**Output**
4	
2 * 7	14
12 + 5	INVALID
Ta-Da	INVALID
8 - 9	-1

E. Minimal Pairs Product Sum

Write a function that receives 5 number of integers and returns the minimal sum of the array (sum of the products and each two adjacent numbers). The challenge is to find the best possible sort of the array elements, to have the minimal sum result. The maximum iteration of sorting is 100.

For Example

Without sorting the array [40, 25, 10, 5, 1], the sum is:

(40*25) + (25*10) + (10*5) + (5*1) = 1305

After do some sorting, the expected output for the array is 225 with sorted array: [40, 1, 10, 5, 25]

Input

First line of input will be 'n' number of test cases, followed by 5 different integers (separated by space) in each line for 'n' lines. Integer input will be in range of 0 <= x <= 50

Output

The output should be the minimal sum result. As shown in the Sample Input / Output table below.

Sample Input / Output

Input	Output
3	
6 4 36 29 42	674
8 3 0 22 38	90
12 31 3 4 21	261

F. Farming

Author: Shahril

Ali has just started to do farming after he went into retirement (like most old folks do). He has selected few plants that he deemed worth for him to plant for. After doing some research online (Ali is a tech-savvy grandpa), he found out that some plant requires combination of special fertilizer for them to grow excellent.

However, there are few special fertilizer products in the market. Also, some fertilizer products also have different amount of quantity, which makes it limited to be applied into some areas.

Ali was thinking about doing simulation. Given few fertilizer which costs differently and can be applied to a limited amount of area, he wanted to calculate the amount of costs (read; money) required for a single plant after certain amount of fertilizers have been applied.

You, as his grandchild, is a programmer. So he asked you to solve this problem, which he promised to give some really good candy if you managed to help him!

Input

The first line consists of integer ***N (1≤N≤10,000)***, which means ***NxN*** area of Ali's farm size.
Second line consists of integer ***L (1≤L≤10,000)***, which means the number of instructions that Ali wants.

The next ***L*** lines consist of char ***P*** and integer ***V***, where P is a single letter denoting instruction and V is value.

If ***P*** is ***"F" (fertilizer)***, then ***V*** has values of integers ***a1, a2 (1≤a1≤a2≤N)*** , ***b1***, ***b2 (1≤b1≤b2≤N)***, and ***fc (1≤fc≤1000)***. ***a1/a2*** is row/column of top-left of rectangle, and ***b1/b2*** is row/column of bottom-right of rectangle. ***fc*** denotes cost of fertilizers for every plant in above mentioned rectangle.

If ***P*** is ***"Q" (query)***, then ***V*** has integers of ***c1*** and ***c2 (1≤c1≤c2≤N)***, which denotes row/column of a cell in Ali's farm. You need to return what is cost of plants in ***c1/c2*** after certain amount of fertilizers have been applied.

Output

For every instruction query instruction, output a line with single integer which is total cost of plant's fertilizer in **c1/c2** location.

Sample Input	**Sample Output**
3 4 F 1 1 3 3 1 F 2 2 3 3 2 Q 1 2 Q 2 3	1 3

Explanation:

After fertilizers have been sprinkled into the farm, here is the visualization of the cost of each plant.

1	1	1
1	3	3
1	3	3

First query asks for cost of location row 1 and column 2, which the value is 1.

Second query asks for cost of location row 2 and column 3, which the value is 3.

G. Floyd's Triangle

Floyd's triangle is a right-angled triangular array of natural numbers, used in computer science education. It is named after Robert Floyd, who was an eminent computer scientist.

It is defined by filling the rows of the triangle with consecutive numbers, starting with a 1 at the top left corner:

```
1
2 3
4 5 6
7 8 9 10
```

Write a program to print the Floyd's triangle and reverse Floyd's triangle by taking the number of rows as input.

For example, the input is 3. The output should be

```
1
2 3
4 5 6
6 5 4
3 2
1
```

Input

The first input is the number of test cases (n), followed by the number of rows for the first top triangle.

Output

Display the combined Floyd's triangle and reverse Floyd's triangle. See example below:

Sample Input / Output

Input	Output
1 2	 1 2 3 3 2 1

H. Find the First Recurring Character!

Write a program that takes a string as input and returns the first recurring character as the output.

Input

First line of input will be 'n' number of test cases, followed by a string.

Output

The output should be the recurred character or if not available, display None. Refer the Test Sample Table below;

Sample Input / Output

Input	Output
3 ABCA BCABA ABC	 A B None

I. Froggie Out!

A frog is at the bottom of *m* meter well. He need to jump until he escape from the well. Each jump takes 1 hour for 1 meter and he will rest and restoring his energy for another 1 hour after the third jump. While he is resting, he will slips for 1 meter backwards. How many days and hours does he take to escape from the well?

Note: The frog will be out of the well if he reach more than *n* meter of the well.

Input

First line will be an integer *n*, which represent the number of cases, followed by *n* number of *m*. The value of *m* must be equal or more than 10 and less than 1000.

Output

Your example output should be in the following format: 2d0h (which d represent days and h represent hours), as shown in the sample below.

Sample Input / Output:

Input	Output
3	
30	2d10h
24	1d22h
15	1d3h

J. Kaprekar's Constant

The number 6174 is known as Kaprekar's constant. It is ultimate convergence point of the Kaprekar's routine, an algorithm thought up by Indian mathematician D.R. Kaprekar in 1949.

The routine is as follows:

1. Take any four-digit number (at least two different digits must be used, zeroes allowed).
2. Arrange the digits in descending and then in ascending order to get two four-digit numbers.
3. Subtract the smaller number from the larger and get the result.
4. Repeat steps 2-4 with the new number.

After a few iterations, the algorithm converges to a fixed point and starts to result in the same number (6174) the so-called Kaprekar's constant.

For Example

For the input of 6324, the solution is as below:
(1) 6432 - 2346 = 4086
(2) 8640 - 0468 = 8172
(3) 8721 - 1278 = 7443
(4) 7443 - 3447 = 3996
(5) 9963 - 3699 = 6264
(6) 6642 - 2466 = 4176
(7) 7641 - 1467 = 6174
(8) 7641 - 1467 = 6174
The number of iteration for the second Kaprekar's constant is 8.

Input

The first input should be the number of cases (n), followed by 4 distinct digits of number for each n line. The input should be in between 1111 <= x <= 9999

Output

The output should be the number of iteration for the second Kaprekar's constant.

Sample Input / Output

Input	Output
3 2619 9321 7319	 3 4 3