Robotique
CRC
Robotics

CRC

PROGRAMMING COMPETITION

KRYPTIK

PRELIMINARY PROBLEM 2

A FEW NOTES

- The complete rules are in section 4 of the rulebook.
- You have until **Sunday, January 28, 11:59 pm** to submit your code
- Feel free to use the programming forum on the CRC discord to ask questions and discuss the problem with other teams. That's what it's there for!
- **We are giving you quick and easy-to-use template files for your code and the tests. Please use them.**

USING THE TEMPLATE FILE

- Basically, the template tests calls the function to test and takes the output and allows you to quickly check if it did what was expected or not. **All your code (except additional functions and classes, which should go right above it in the file) should be written in the function of the part of the problem you are solving.**
- Points given in the document are indications of how difficult the section is and how many points you will get if you complete it. This preliminary problem is going to be 2% of the main challenge towards the global score of the programming competition and for more points related information consult the rulebook.

STRUCTURE

Every problem contains a small introduction like this about the basics of the problem and what is required to solve it. Points distribution is also given here for the preliminary problems.

Input and output specification:

In these two sections, we specify what the inputs will be and what form they will take, and we also say what outputs are required for the code to produce and in what format they shall be.

Sample input and output:

In these two sections, you will find a sample input (that often has multiple entries itself) in the sample input and what your program should produce for such an input in the sample output.

Explanation of the first output:

Sometimes, the problem might still be hard to understand after those sections, which is why there will also be a usually brief explanation of the logic that was used to reach the first output from the first input.

Secret Cod(ing challenge)

We want to establish a secret code to transmit messages, but as this document is public, it's your programming skills that will enable us to transmit hidden messages. Your task is to establish 4 keys with the first 4 parts and then encrypt the message in the last part.

Part 1: Tools for success (30 points)

You have a lot of tools in your workshop, and you don't want to count them one by one. To count faster, you make small groups of tools. Groups of 5, 7, 23, whatever prime numbers come to mind.

However, when you do your groupings, there are still a few tools that aren't in any groups. Be happy in the knowledge that it's thanks to these remaining tools that you'll know the minimum number of tools in your possession!

So you'll have different groupings and leftovers to help you find the minimum number of tools. You'll need the number of tools in each grouping, which we'll name g_1, g_2, g_3, … and also the number of tools left over after grouping by pack of g tools. These remaining tools will be named r_1, r_2, r_3, …

The final set of numbers important to the formula are the minimum number to multiply all other g groupings to obtain one more than a multiple of the grouping. We'll call these numbers m_1, m_2, m_3, … Here's the formula to calculate m_1:

$$m_1 * (g_2 * g_3 * g_4 * \ldots) \equiv 1 \ (mod\ g_1)$$

In short, as all the numbers of tools per grouping g are given, we need to find the smallest m_i for which all the other g multiplied together give 1 more than a multiple of g_i.

Once the number m has been calculated for each grouping, we can apply the formula to find a possible number of tools x.

$$ProdG = g_1 * g_2 * g_3 * \ldots$$

$$x = (ProdG/g_1 * r_1 * m_1) + (ProdG/g_2 * r_2 * m_2) + (ProdG/g_3 * r_3 * m_3) + \ldots$$

The x we found is a possible number of tools, but not necessarily the smallest. If we want to get the smallest one, we need to apply a modulo of the product of all g to the x we just found.

$$min = x\, mod(ProdG)$$

Input specification:

As an input you will receive an array of tuples containing the amount of tools in the groupement(g) and the number of tools left out(r) using the groupement g. The format is:

$[(g_1, r_1), (g_2, r_2), (g_3, r_3), (g_4, r_4), \ldots]$

Output specification:

You need to return the minimal number of tools called min as a string filled with zeroes at its beginning to give a 4-digit key.

Sample input:

```
[(3,2), (5,3), (7,2)]
[(21, 5), (2, 1), (19, 7), (5, 2)]
[(7,4), (11,2), (23,14), (3,2)]
```

Sample output:

```
0023
3617
5051
```

Explanation of the first output:

We know that we have 2 tools left over when we make groups of 3, we have 3 tools left over when it's groups of 5 and, finally, 2 tools left over in our grouping by groups of 7 tools.

So we have $g_1, g_2, g_3, r_1, r_2, r_3$ and now need to find the m for each of the groupings. Let's find the m for the grouping by groups of 3 with the formula:

$$m_1 * (g_2 * g_3 * g_4 * ...) \equiv 1 \ (mod\ g_1)$$

We want to find the smallest positive m for which all other g multiplied and then modulo g_1 gives 1.

If we replace the numbers we know and try with $m_1 = 1$

$$1 * (5 * 7)\ mod(3) \equiv 2$$

The result of the modulo we want is 1 and we get 2. The m_1 we tried is therefore the wrong one. So we try $m_1 = 2$

$$2 * (5 * 7)\ mod(3) \equiv 1$$

This time the modulo result is 1, so we keep $m_1 = 2$.

Once the process has been completed for the 3 groups, we find $m_1 = 2$, $m_2 = 1$, $m_3 = 1$.

We can then apply the formula to obtain x

$$x = (ProdG/g_1 * r_1 * m_1) + (ProdG/g_2 * r_2 * m_2) + (ProdG/g_3 * r_3 * m_3) + ...$$

$$x = (105/3 * 2 * 2) + (105/5 * 3 * 1) + (105/7 * 2 * 1)$$

$$x = 140 + 63 + 30$$

$$x = 233$$

With our x, we can now complete the problem and find the minimum number of tools, called min

$$min = x\ mod(ProdG)$$

$$min = 233\ mod(105)$$

$$min = 23$$

We have a minimum of 23 tools for the first example!

Part 2: One small step for man, one giant leap for… NUMBERS!?? (35 points)

Here, you'll need to familiarize yourself with literally extraterrestrial arithmetic: lunar numbers! Lunar numbers have the property of taking the maximum and minimum of two digits as their principal operations. Thus, addition and multiplication in this algebra are defined as follows:

$$a + b = max(a, b)$$
$$a \times b = min(a, b)$$

So, 2+7=7 but 2*7=2. But beware, as the operations are performed directly between digits, you need to check that the plus-size operations still work. As addition is already a pairwise operation, everything's fine and, comparing columns, 956+379=979, for example. However, multiplication is distributive between the digits of two numbers. Thus:

$$811 * 81$$
$$= 111 + 8110$$
$$= 8111$$

As with normal multiplication written in columnar form, you actually have to multiply 811 by 1 first, and then by 80. Adding up the respective digits, you end up with 8111, not 811, as we might have naively expected from our definition of multiplication. So you'll have to be careful when multiplying.

The main advantage of this wacky algebra is that it doesn't contain any remainder to be carried forward to the next digit when multiplication in column form is used. For example, 4*7 gives 8 to the unit, but would require an extra 2 to be added to the beginning of the number in normal algebra. Here, in lunar arithmetic, the *digits* of the answer can theoretically be calculated independently of each other.

Input specification:

As input, you'll receive a string composed of two positive integers separated simply by a lunar arithmetic operation (simply + or *).

Output specification:

As the output, you'll have to return the result of the requested calculation according to the bizarre laws of lunar arithmetic.

Sample input:

```
811*81
345+4567
57*557
```

Sample output:

```
8111
4567
5557
```

Explanation of the first output:

See the explanation in the problem’s description.

Part 3: Where’s Waldo!? (15 points)

We all know how to find Waldo, but he's just disguised himself as a number! He's gone into hiding among a long sequence of numbers, and we'll have to find him before he can finally answer for his actions. Fortunately, our informant knows which number he's disguised as, so all we have to do is find it in a "scene". For a given number and sequence of digits, simply indicate the position of the last digit of the number found (Waldo’s last digit) to obtain the key.

Input specification:

As input, you'll receive a string containing a little bit less than 10,000 digits randomly separated by other characters, as well as a 9-digit positive integer (Waldo is as long as 9 digits!) to be found in the sequence.

Output specification:

The output should be a positive integer corresponding to the position of Charlie's last digit. In the unlikely event of multiple occurrences, only the first occurrence should be considered.

Sample input:

```
The beginning and end of this input are truncated to just show an
example and the real example isn’t highlighted.

big_string:
…0324k39460492499123456789565423105781086579872507889213941606121x031
5198246031562446000217102554251380531600291O…

sequence_to_find:
123456789
```

Sample output:

```
position: 1579
```

Explanation of the first output:

In this truncated sequence, the last character of the sequence to find is located at the 1579th digit of the input string (starting the count with the first character being 0).

Part 4: Basically (40 points)

The aim of this section is to establish a protocol for converting any number into a 4-digit key. The first transformation step is to change the number in base 10 to a number in another base, then express the digits in base 10 to form a new concatenated number and transform it further. You'll need to do this step 5 times. For example, let's take base 29 and the number 982:

$$(982)_{10} \Rightarrow (14P)_{29} \Rightarrow 1425 \text{ (P correspond à 25)}$$
$$(1425)_{10} \Rightarrow (1K4)_{29} \Rightarrow 1204$$
$$(1204)_{10} \Rightarrow (1CF)_{29} \Rightarrow 11215$$
$$(11215)_{10} \Rightarrow (D9L)_{29} \Rightarrow 13921$$
$$(13921)_{10} \Rightarrow (GG1)_{29} \Rightarrow 16161$$

The first step would therefore encode 982 to 16161. Then, the second step consists in classifying the digits of the resulting number according to the result of a modulo 4. So, following the same example:

$$1 \equiv 1 (mod\ 4)$$
$$2 \equiv 6 (mod\ 4)$$

Finally, to find the key, we need to add up all the digits of the same modular congruence group several times until only one digit between 0 and 9 remains. So, with 16161:

0 (groupe 0 mod 4)
1+1+1=3 (groupe 1 mod 4)
6+6=12, 1+2=3 (groupe 2 mod 4)
0 (groupe 3 mod 4)

The final concatenation of these 4 digits gives the key 0330 that we will return in a string to keep the formatting.

Input specification:

As input, you'll receive two positive integers, the first corresponding to the conversion base (between 2 and 99, you don't need to express the converted number in the base with letters, this was just to demonstrate an intermediate step), and the second corresponding to the input number to be encoded.

Output specification:

As output you will need to return a string representing the 4-digits key.

Sample input:

```
29 982
23 1053
65 812
```

Sample output:

```
0330
8400
0707
```

Explanation of the first output:

See explanation of the problem.

Part 5: Advanced notions in cryptography (25 points)

All the numbers you've got are useful! You and your business partner have established an elaborate means of confidential communication using this problem, but there's one last layer of security needed to top it all off. You'll need to encrypt the message using the RSA encryption protocol.

First, you need to differentiate between the numbers that will serve as part of the key and the number that will be the message. **RSA encryption requires two prime numbers p and q.** Using these two primes, it is possible to construct two other highly useful numbers:

$$n = p * q$$

$$\varphi(n) = (p - 1) * (q - 1)$$

That's enough for the two prime numbers. Next, you'll need to get **a number that is coprime with (and less than)** $\varphi(n)$. Remember that two numbers that are coprime with each other are two numbers that share no divisors except 1. For example, 4 is prime with 7, but 4 is not simply prime. This number is called the cipher exponent and will be denoted by the variable e. **The last number is therefore, by elimination, the message to be transmitted M. The message will be encoded as follows:**

$$C \equiv M^{e} \ (mod\, n)$$

where C is the encrypted message and mod is the modulo operation. All that remains is to check that it is possible to decrypt the message with the information also obtained from parts 1 to 4. We'll need to verify the calculation by finding the decryption exponent written d. d is the modulo $\varphi(n)$ inverse of e:

$$d * e \equiv 1 \ (mod\, \varphi(n))$$

There are several existing methods for calculating this decryption exponent. The first is that, by the properties of modulo, we must find a pair of integers u and v that satisfies:

$$e * u + \varphi(n) * v = 1$$

Such a pair of integers can be found by iteration or with the extended Euclid algorithm. It's up to you to find the methodology of your choice! Of this pair, u corresponds to the modular inverse of e and will have to be reduced to its simplest form to correspond to d. For example, if we have e=4 and (n)=7, u can be -5, 2, 9, 51, and so on. However, only 2 falls within the "main" interval from 0 to 7, so only 2 will be accepted as d. To decrypt the message, simply perform the following operation:

$$M \equiv C^{d} \ (mod\, n)$$

When in doubt about being able to evaluate such large numbers, it's worth remembering that it will always be possible to evaluate the whole thing by computer, thanks to the following modulo property:

if $a \equiv b \ (mod\, n)$, then $a^{k} \equiv b^{k} \ (mod\, n)$

It is therefore possible to evaluate the exponential expression "progressively". **Your computers should be able to process C quite easily without this. However,**

this will likely not be the case with the reverse operation of getting back M to check the validity of d if you want to use it.

Clarification on RSA: this is not exactly how the RSA encryption algorithm is usually used, but it gives a good introduction to the subject. (e, n) is the public key and (d, n) is the private key. Normally, the person sending a message doesn't know the prime numbers p and q, so it's not as simple to determine d, which would be a security problem. In theory, only the "recipient" knows these two numbers, which are usually huge. The algorithm uses the great challenge of the prime number factorization problem to establish a public key that reveals very little about the private key. **If you'd like more information on certain aspects of this problem, RSA encryption, Euclid's algorithm and modular algebra are all fairly well known topics.**

Input specification:

As input, you'll receive 4 strings corresponding to the 4 distinct integers p, q, e and M. (in the same order as the one given by the individual parts!)

Output specification:

As output, you'll need to give a tuple containing an integer corresponding to the encrypted message C, followed by an integer corresponding to the decryption exponent d.

Sample input:

```
p:0023, q:8111, e:1579, M:0330
p:3617, q:4567, e:3517, M:8400
p:5051, q:5557, e:3833, M:0707
```

Sample output:

```
(168442, 127459)
(13038683, 8379733)
(26895292, 16016297)
```

Explanation of the first output:

Using p and q, we first find n=186553 and (n)=178420. Calculating the encrypted message then gives C=168442 and modular inversion gives d=127459.