Nuxtu SAS
Technical examination
Software Engineer - Low-level programming
01/Oct/2021
Deadline: 04/Oct/2021 - 14:00

For this technical examination you may use any books, notes, web pages, tutorials, documentation, or any other type of information source. You **may not** discuss this exam or questions related to the exam with your fellow contestants. It is also prohibited to ask for help from anyone you know that has experience on these topics, doing so will disqualify you from this and any future selection processes in *Nuxtu*.

Feel free to ask any question related to this examination to our *Chief Technology Officer* through WhatsApp or phone call at *+57 (301) 658 - 3977*, remember: **the only bad question is an unasked one**.

At the end of this examination, the following deliverables are expected:

- A forked GitHub repository from this original repository, with continuous commits on your progress. **Your last commit before the deadline will be evaluated**.
- **Two** well-documented C++ programs (.cpp files) that solve the proposed challenge. Code complexity and design, execution time, coding style and resilience will be taken into account during the grading process.
- Be prepared to answer any questions related to your solution. This will show us that you completely understand your implementation, which is the most important part of this examination.

Work hard. Have fun. Make history!

Jeff Bezos

Your challenge will be to create an RSA encryption and decryption program. RSA encryption allows people to exchange information securely over the internet nowadays by the use of a **public key** and a **private key**. (Further reading on RSA encription: https://brilliant.org/wiki/rsa-encryption/).

1 Creation of a RSA key pair - Description of the algorithm

You start with a pair of prime numbers p and q. Let's say that:

$$p = 13$$
$$q = 17$$

In general, p should always be 13 or greater, and q should always be 17 or greater.

The number n is defined as $n = p \times q$. In this case, $n = 13 \times 17 = 221$. The function $\phi(n) = (p-1)(q-1)$, which in our example is equal to $\phi(n) = 12 \times 16 = 192$.

Now we need another integer e that is greater than 1 and less than $\phi(n)$ It is important that $\phi(n)$ is not divisible by e. A good value for e in this example could be 5, because it is greater than one, is less than $\phi(n)$ and does not perfectly divide 192. A good idea is to choose an e that is small.

Your **public key** is two values: n and e. Here, your public key is the value 221 together with the value 5.

To calculate the private key d, we use this formula:

$$d = (i \times \phi(n) + 1)/e$$

Where i is any integer. For the sake of simplicity, you can use $i = 2$.

$$d = (2 \times 192 + 1)/5 = (384 + 1)/5 = 385/5 = 77$$

In the end:

- Your public key is the numbers $n = 221$ and $e = 5$
- And your private key is $d = 77$

2 Encrypt.cpp - Encrypting using a public key

The cipher c (the encrypted version of the text) is calculated like this:

1. You assign each letter of the message a number. For example, you can use each letter's position in the alphabet.
2. You apply the encoding formula (described later) to the resulting sequence of numbers.

You can use this replacement dictionary for your letters (use only capital letters for simplicity):

Letter	Code
A	1
B	2
C	3
D	4
E	5
F	6
G	7
H	8
I	9
J	10
K	11
L	12
M	13
N	14
O	15
P	16
Q	17
R	18
S	19
T	20
U	21
V	22
W	23
X	24
Y	25
Z	26
Space	27

The formula to calculate c is (for each character):

$$c = translation^e \mod n$$

For example, for the letter P, this would lead (in our example) to:

$$c = 16^5 \mod 221 = 152$$

The program requirements are as follows:

- The program must request the following parameters from the user via terminal (*cin*):
  - Value of n.
  - Value of e.
  - String to encode.
- The program must return (via terminal) a list of numbers separated by comma that represent the encoded version of the string. Use the provided table for the encoding (all capital letters).

3 Decrypt.cpp - Decryption using a private key

To decrypt a cipher c, you simply need to use this formula:

$$decrypted = c^d \mod n$$

For our previously encoded letter P, this would lead to:

$$decrypted = 152^{77} \mod 221 = 16$$

The program requirements are as follows:

- The program must request the following parameters from the user via terminal (*cin*):
  - Value of n.
  - Value of d.
  - String of numbers (separated by comma) to decode.
- The program must return (via terminal) the original phrase that was encoded.