Instructions

- This assignment is due on Tuesday, November 17, 2020 at 2:00 PM EST. Late submissions will **not** be accepted.
- This assignment consists of two problems. You should choose one for submission.
- Your solution needs to be formatted using the LaTeXtemplate available on OWL.
- All solutions must be written in full sentences.
- You are not allowed to work with others or use any online resources.
- This assignment is worth 5 points.

Problem 1.

In class, we investigated quadratic residues modulo a prime. In this problem, we will take it up a notch by investigating *cubic* residues modulo primes.

We begin by fixing a prime $p > 3$. Furthermore, let a be an integer such that $p \nmid a$. We say that a is a *cubic residue mod* p if there exists b such that $a \equiv b^3 \pmod{p}$.

1. Show that the product of two cubic residues mod p is again a cubic residue.
2. For a and b not divisible by p, is it necessarily the case that at least one of: a, b, ab is a cubic residue mod p?
3. Fix a primitive root $g \in \mathbb{F}_p^*$. Determine for what values of k, the element g^k is a cubic residue mod p.

   Hint. Your answer should depend on what p is mod 3.

Problem 2.

Most of the cryptosystems implemented in practice add an element of randomization in order to prevent certain attacks. In this problem, we will explore one way of adding randomization to the RSA cryptosystem.

Our set of messages $M = \{0,1\}^{30}$ consists of bit strings of length 30. Alice's public key is the usual RSA public key (N, e). To encrypt a message m, Bob chooses a random bit string r of length 30 and computes

$$c = n(m)^e \pmod{N}$$

where $n(m)$ is the number whose binary representation is given by the bit string $r||(r \oplus m)$. Here, $||$ denotes concatenation of strings and $\oplus$ is a binary operation defined on individual bits as:

$$b \oplus b' = \begin{cases} 1 & \text{if } b = b' \\ 0 & \text{otherwise.} \end{cases}$$

You task is to implement the decryption function that given the RSA private key finds the original message m.

Statement

The assignment has two parts.

1. Write a function in Python3 called `solve` that, given an RSA private key (N, d) and a ciphertext c produced by the above algorithm above, finds its decryption $m \in \{0,1\}^{30}$.
2. Download the file `generate_input.py` from OWL, use it to obtain three sets of inputs, each set consisting of an RSA private key (N, d) and a ciphertext c, by running

   ```
   python generate_input.py [last 3 digits of your student number]
   ```

   and run your program on these three inputs.

Your submission must consist of a single PDF file containing:

1. the *Python code* implementing your solution;
2. and the three *inputs you generated*, and the *output of your program* run on these three inputs.

Examples

Here is an example of what your function `solve` should do:

```
>>> solve(1152924549740517683,1152924547593031199,3861264878320009 48)
Decrypted message m =  11100011111001110000011010 0011
>>> solve(1152923261248820939,1152923259101335655,5988371857280778 36)
>>> solve(2,[1,1,0,1],[1,1,0],[1,1,1])
Decrypted message m = 11011000111100010110101011 0110
>>> solve(1152921753714962639,1152921751567478759,6129698803913946 94)
Decrypted message m = 00010101101110100010101111 0010
```

Notes

- The file `generate_input.py` is written in Python3, and so should be your solution. Make sure you are using a 64bit version of Python3
- Your submission must use the LaTeX template available on OWL.
- Your code should not make use of any external libraries such as `numpy` or `math`. All the auxiliary functions should be implemented by you, and should be included in your submission. You should only use the most basic arithmetic operations such as `+`, `-`, `*`, `//`, `%`.
- Comments in the code are not mandatory. However in the case of an incorrect solution, the comments can provide grounds for partial credit.