The code for this TP contains three files (`tp_code.py`, `blackbox.py`, `aes.py`). The file `tp_code.py` contains the following functions:

- `base64_encode` and `base64_decode` convert between a list of integers representing bytes (values between 0 and 255) and a string in Base64 (64 characters including alphanumeric and "+/")
- `weakCipher1` and `weakCipher2` are two ciphers

The file `blackbox.py` is obfuscated. Importing it will give you access to three functions:

```
from blackbox import weakCipher3, weakCipher4, weakCipher5
```

Each of them takes as input a list of 16 integers representing bytes, and return a similar list (the result can then be converted to a string using `base64_encode`)

We start by focusing on the ciphers in `tp_code.py`.

Question 1. *You intercepted the following ciphertext:*

```
,?jtajY.tcrL0tvtl?20.Y
```

The encryption procedure is the function `weakCipher1` *and it seems that the 3 first letters actually correspond to the word* `You`*. Decrypt the whole message and retrieve the key.*

Question 2. *You intercepted the following ciphertext:*

```
j5YTlG!wLkVu2QU 5,Iz.QlGApwkUGEYlL0,TzYiC3m0lYhtE,gzGu7TCc
```

The encryption procedure is the function `weakCipher2` *and it seems that the first word is* `This` *and the last one* `points!`*.*

Decrypt the whole message and retrieve the key.

Question 3. *You intercepted the following ciphertext:*

```
0liLBTrm2?B.rsBzr7u 2K2ZB0iLBE.gBz.8iL2Y8YD0B.rsB,TIQ7oFmMl1AJBrrUoQDz
```

The encryption procedure is the function `weakCipher2` *and it seems that the plaintext contains the word* `challenge`*.*

Decrypt the whole message and retrieve the key.

We now focus on the ciphers in `blackbox.py`.

Question 4. *Consider the function* `weakCipher3`*.*

1. *Pick one random message M and encrypt it.*
2. *Modify one byte of M and encrypt the new message.*
3. *Compare both ciphertexts byte by byte. What do you observe?*
4. *Decrypt the following ciphertext encoded in base64:*

`9GVHL2Jb+QLfityW1Umw5w==`

Question 5. *Condider the function* `weakCipher4`*. Recall that to take the XOR of two bytes (represented as integers here), you can use the operator* `^` *in Python.*

1. *Pick two random messages M_1 and M_2 and encrypt them.*

2. *Pick a random message M_3 and let $M_4 = M_1 \oplus M_2 \oplus M_3$. Encrypt them.*
3. *Compare $C_1 \oplus C_2$ and $C_3 \oplus C_4$ where C_i is the ciphertext corresponding to M_i.*
4. *Let $M_1 \oplus M_2 = a \parallel b$ with $|a| = |b| = 64$ bits. Verify that $C_1 \oplus C_2 = b \parallel a \oplus b$.*
5. *What is the inverse of the application $(a, b) \to (b, a \oplus b)$?*
6. *Decrypt the following ciphertext encoded in base64:*

`rv6mp36Doa6Zyt2WjMDd6w==`

The file `aes.py` contains an implementation of the block cipher AES. It can be used as follows:

- define an object `a = AES(k)` where k is a list of 16 bytes (as integers);
- use `a.encrypt(l)` or `a.decrypt(l)` to encrypt (resp. decrypt) an AES block represented as a list of 16 bytes (as Python integers between 0 and 255).

Question 6. *Consider the function* `weakCipher5`*. You are informed that* `weakCipher5` *is a double AES encryption, as follows:*

```
return AES(k2).encrypt( AES(k1).encrypt(l) )
```

where l is the input 16 bytes. Furthermore, you are informed that both `k1` *and* `k2` *have only their two first bytes which are non-zero.*

Decrypt the following ciphertext encoded in base64:

`ZTCG1sNoSzT415YbsiZ+yw==`