RC4

1. Step 1: program your own RC4

From the description on Wikipedia code your own version of RC4 and test Roos bias. This toy implementation will help you to answer the other questions.

2. Step 2: funny versions of RC4

In many implementation of RC4, the implementation of $\texttt{swap}(S[i_{t+1}], S[j_{t+1}])$ is often coded by:

```
tmp=S[i]
S[i]=S[j]
S[j]=tmp
```

However, it requires 3 variables (and thus 3 registers in CPU). A well-known trick consists to use the xor operation:

```
S[i]^=S[j]
S[j]^=S[i]
S[i]^=S[j]
```

Question 1: Compare two implementations of RC4: a classical one and other implementation using the xor trick. Chat do you observe ?

Question 2: What are the consequences on the initialization of RC4 ?

Question 3: What are the consequences on the keystream produced by RC4 ?
Question 4. Let consider the full implementation of RC4 but we modify the keystream generation as follows:

```
int i=0,j=0,x,t;
for (x=0; x < len; ++x)
{
  i = (i + 1) % 256;
  j = (j + state[i]) % 256;
  z[x] = state[(state[i] + state[j]) % 256];
}
```

Let denote i_t, j_t and z_t the respective values of i, j and of the keystream after t rounds.
Give formula for i_{t+256}, j_{t+256}, j_{t+512}, z_{t+256} and z_{t+512}. What do you observe ?

3. Step 3: the first step of FMS

Let $S_j[i]$ be the value of the i^{th} byte of the permutation S after the j^{th} round. Let denote z_j the byte of the keystream after j rounds.
Question 1. Show that if $S_0[2] = 0$ and $S_0[1] \neq 2$ then $z_2 = 1$.

Question 2. Demonstrate that the distribution of z_1 is not uniform (compare when S is initialized by a random permutation).

Question 3. What can you conclude ?