Task: Stream Cipher implementation:

Input a key K - 64 bits and message M - of n bits to yield output C - also n bits.

(The code should be able to handle n upto $(2^{64} - 1)$. The key is 8 bytes (64 bits), to be read from a binary file Key. The file should have at least 8 bytes and you read the first 8 bytes (even if file is longer).

The plaintext is also to be read from binary file, Input (it could be say a mp3 file or a jpg file or a text file).

The output - ciphertext is to be written to a binary file Output. All the file input - output should be done in bytes. The bits in a byte are to be read left to right (just as a convention).

The names of Input Output and Key files should NOT be hardcoded but passed on the command line

The command should be
$ python stream.py Input Output Key
or for C++, if the executable is named stream
$ stream Input Output Key

The decryption will be achieved by the same command:
$ python stream.py Output decoded_input Key
$ stream Output decoded_input Key

The streamcipher is to be generated by combining 3 LFSRs along with a nonlinear combining function :

3 LFSR's with $m = 16$, $n = 17$, $k = 31$ ($m + n + k = 64$)
and primitive polynomials:
$p_1(x) = x^{16} + x^5 + x^3 + x^2 + 1$,
$p_2(x) = x^{17} + x^3 + 1$
$p_3(x) = x^{31} + x^3 + 1$.
Take the combining function to be

$$f(w, u, v) = v \otimes w + (1 \ominus v) \otimes u.$$

(w corresponds to p_1, u to p_2 and v to p_3).

The parameters m, n, k and the polynomials are to be hardcoded in the program file and not passed on command line, but you can test the code using different values:

Details:

From the keyfile - the code should read first 64 bits $K : k_0, k_2, \ldots, k_{63}$ (8 bytes) and out of that take $w_0, w_1, \ldots, w_{15}$ to be the first 16 bits, $u_0, u_1, \ldots, u_{17}$ to be the next 17 bits and let $v_0, v_1, \ldots, v_{31}$ be the last 31 bits:

$$w_j = k_j, \quad u_i = k_{15+i} \quad v_t = k_{32+t}$$

$0 \leq j \leq 15$, $0 \leq i \leq 16$, $0 \leq t \leq 30$.

If the key file has ABCDEFGH..., then the first 8 characters are ABCDEFGH and in ASCII/ UTF-8 code will be 65,66,67,68,69,70,71,72 - in bits it would read: (for easy counting, I am alternating colours after 8-bits)
01000001010000100100001101000100010001010100011001000111010010000

Thus, $w_0, w_1, \ldots, w_{15}$ would be 0100000101000010
$u_0, u_1, \ldots, u_{16}$ would be 01000011010001000
and $v_0, v_1, \ldots, v_{30}$ would be 10001010100011001000111010010000

Recall :

Given a polynomial

$$p(x) = 1 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1} + a_m x^m$$

with $a_1, a_2, \ldots, a_{m-1}, a_m \in \{0, 1\}$ and initial values $s_0, s_1, \ldots, s_{m-1} \in \{0, 1\}$, the output of the LFSR s_j for $m \leq j \leq n$ is given by

$$s_j = (a_1 s_{j-1} + a_2 s_{j-2} + \ldots + a_m s_{j-m}) \text{ modulo } 2, \quad j \geq m$$

You can test your code for LFSR with the following data:

For $m = 8$,
if the coefficients of the primitive polynomial $\{a_1, a_2, \ldots, a_8\}$ are given by 0,1,0,1,1,1,1,1

and the initial state of LFSR $\{s_0, s_1, \ldots, s_7\}$ are 1,1,1,0,1,0,1,0

then the output $\{s_j : 0 \leq j \leq 31\}$ is:

1,1,1,0,1,0,1,0,1,1,0,1,1,0,0,1,0,0,0,1,0,0,1,0,0,1,0,1,1,1,1,1

The plaintext message M read from the input file be a bitstream of length n-bits:

$$M = m_0 m_0 \ldots m_{n-1}$$

Using the initial values and polynomials specified, generate $\{w_j\}, \{u_j\}, \{v_j\}$ and then $r_j = f(w_j, u_j, v_j)$:

$$r_j = v_j \otimes w_j + (1 \ominus v_j) \otimes u_j, \quad j < n.$$

Then obtain the cipher text

$$c_j = m_j \oplus r_j, \quad 0 \leq j < n$$

The cipher text is to be written to a binary file.

If the message M read from the Input file begins with Agreed, which in bits would be 01000001011001110111001001100101010110010 1 and if the bitstream coming from combination of LFSR's is
00100101010001110011011000110111011100011 then the bits of the ciphertext (obtained by bitwise XOR, i.e. addition modulo 2) will be
01100100001000000100010001010010000101000

01000001011001110111001001100101010110010 1
00100101010001110011011000110111011100011

01100100001000000100010001010010000101000

I suggest that each of you checks their own outputs with at least two other friends before you submit. Last date for submission : 19 Feb (by 2355).