Problem 2. «AntCipher 2.0»

Sam studies microelectronics, while his hobbies are biology and cryptography. He united all these areas in a research project aimed at constructing a tiny GPS tracker for an ant to monitor its movements. When coordinates are determined, they are encrypted and transmitted to a Sam's computer, where they are automatically decrypted. Sam developed a symmetric cipher AntCipher for this purpose, but it was quite weak. That is why Sam developed a new symmetric stream cipher called AntCipher 2.0.

Once a minute, the tracker determines its GPS coordinates using satellites. Then the latitude as an IEEE 754 single-precision floating-point value is converted into a 32-bit binary sequence, while the same is done with the longitude. These two sequences are concatenated (latitude $\|$ longitude) to form a 64-bit plaintext. The plaintext is bitwise XORed with a keystream produced by the cipher thus forming a 64-bit ciphertext which is transmitted to the computer.

The cipher works as follows. At the initialization stage, a 64-bit secret key is written to a 64-bit register R. In iteration number $i, i \geq 1$, a 64-bit sequence (keystream) K_i is produced taking a value of R as an input. The keystream is also used to update the register: at the end of the iteration, K_i is written to R. Consider the following CNF C, where CNF is a conjunction of disjunctions of literals, yet literal is a Boolean variable or its negation:

$$C = (x_1 \vee x_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_5) \wedge (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3 \vee x_5) \wedge (x_2 \vee x_3 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_3 \vee x_5) \wedge (x_1 \vee x_2 \vee \neg x_6) \wedge (\neg x_1 \vee \neg x_2 \vee x_6) \wedge (x_1 \vee x_4 \vee \neg x_6) \wedge (\neg x_1 \vee \neg x_4 \vee x_6) \wedge (x_2 \vee x_4 \vee \neg x_6) \wedge (\neg x_2 \vee \neg x_4 \vee x_6) \wedge (x_1 \vee x_3 \vee \neg x_7) \wedge (\neg x_1 \vee \neg x_3 \vee x_7) \wedge (x_1 \vee x_4 \vee \neg x_7) \wedge (\neg x_1 \vee \neg x_4 \vee x_7) \wedge (x_3 \vee x_4 \vee \neg x_7) \wedge (\neg x_3 \vee \neg x_4 \vee x_7) \wedge (x_2 \vee x_3 \vee \neg x_8) \wedge (\neg x_2 \vee \neg x_3 \vee x_8) \wedge (x_2 \vee x_4 \vee \neg x_8) \wedge (\neg x_2 \vee \neg x_4 \vee x_8) \wedge (x_3 \vee x_4 \vee \neg x_8) \wedge (\neg x_3 \vee \neg x_4 \vee x_8).$$

The equation $C = 1$ represents a nonlinear function F_C that takes a 4-bit input x_1, x_2, x_3, x_4 and produces a 4-bit output x_5, x_6, x_7, x_8. In the i-th iteration of the cipher, a 64-bit value of R is divided into 16 4-bit sequences, which are given to F_C as inputs. Then 16 4-bit outputs are produced and concatenated thus forming a 64-bit K_i that is written to R and is used as a keystream.

On the computer, the cipher is initialized by the same secret key, so the same keystream is produced as on the tracker. When a 64-bit ciphertext is transmitted from the tracker, the corresponding keystream is produced and bitwise XORed with the ciphertext thus obtaining the plaintext. The first 1 704 ciphertexts were transmitted with no problem and the coordinates were automatically decrypted. Then a hard disk drive failure happened on the computer and as a result the secret key, as well as almost all 64-bit ciphertexts and keystreams were lost. Sam could recover only the last 1704-th ciphertext: 1001 1000 0011 1101 0110 0011 1101 0101 1011 0011 1011 0111 0000 0000 1000 0011. Also, the keystreams generated in iterations 1702 and 1703 were partially recovered ('X' stands for an unknown bit value):

- K_{1702} = 0101 1001 1111 0011 00X1 X111 1X00 00X0 111X X000 XXXX XXXX XXXX XXXX XXXX XXXX;
- K_{1703} = XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX X111 000X X010 01X1 0X10 0101 0000 1111.

Please, help Sam to find the plaintext in iteration 1704 to find the ant.