Problem 1. «A conundrum»

Can you crack a conundrum?

b tn ztwobfc twxfc t hutek vptbwbfc t svbeo hukbfq nu vx ntpo xlv
wbfus b ztwo 6 nuvpus fxpvk vkuf 2 nuvpus zusv 5 nuvpus utsv 6 nuvpus sxlvk
3 nuvpus utsv 6 nuvpus fxpvk 4 nuvpus utsv 3 nuvpus sxlvk 3 nuvpus zusv
3 nuvpus fxpvk 6 nuvpus sxlvk 3 nuvpus fxpvk 4.24 nuvpus sxlvkutsv
3 nuvpus utsv 1 nuvpu zusv 6 nuvpus fxpvk 1 nuvpu zusv 3 nuvpus utsv
6 nuvpus sxlvk 6 nuvpus fxpvk 6.49 nuvpus sxlvksxlvkutsv 6 nuvpus fxpvk
4 nuvpus utsv 3 nuvpus zusv 6 nuvpus sxlvk 3 nuvpus utsv 3 nuvpus fxpvk tfq
1 nuvpu zusv zktv bs vku ftnu vktv b ktau zpbvvuf bf vku stfq?

Above is the conundrum sent by Alice to Bob. Find an answer to Alice's question!

Problem 2. «Let's find permutations!»

A function F from $\mathbb{F}_{2^n}$ to itself is called **APN** (**almost perfect nonlinear**) if for any $a, b \in \mathbb{F}_{2^n}$ with $a \neq 0$ the equation $F(x) + F(a + x) = b$ has at most 2 solutions. APN functions possess an optimum resistance to differential cryptanalysis and are under the extreme interest in cryptography! For example, when the unique 1-to-1 APN function in 6 variables was found in 2009, it was immediately applied in construction of the known lightweight cipher FIDES.

Let $F(x) = x^d$. It is known that F is APN for the following exponents d:

- $d = 2^{2i} - 2^i + 1$, $\gcd(i, n) = 1$, $i \geqslant 2$;
- $d = 2^t + 3$, $n = 2t + 1$;
- $d = 2^t + 2^{t/2} - 1$ for t even and $d = 2^t + 2^{(3t+1)/2} - 1$ for t odd with $n = 2t + 1$;
- $d = 2^{2t} - 1$, $n = 2t + 1$;
- $d = 2^{4i} + 2^{3i} + 2^{2i} + 2^i - 1$ with $n = 5i$.

Q1 Problem for a special prize! Describe (characterize or make a list of) all linear functions L_1 and L_2 for any one exponent above for $n = 7$ or $n = 8$, such that the function $L_1(x) + L_2(F(x))$ is a permutation.

Q2 Problem for a special prize! Consider any of the exponents d above. Find linear functions L_1 and L_2 (both different from 0 function) such that the function $L_1(x) + L_2(F(x))$ is a permutation ($n \geqslant 9$), or prove that such functions do not exist.

Remark. Let us recall the following definitions:

- $\mathbb{F}_{2^n}$ is the finite field of order 2^n;
- a function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ has the unique representation $F(x) = \sum_{i=0}^{2^n-1} c_i x^i, c_i \in \mathbb{F}_{2^n}$;
- the algebraic degree of F is equal to the maximum binary weight of i such that $c_i \neq 0$;
- a linear function L has degree at most 1 and $L(0) = 0$ (that is $L(x) = \sum_{k=1}^{n} c_k x^{2^k}$).

Problem 3. «Shuffle ballots»

In electronic voting, n voters take part. Each of them is assigned a **unique identifier** that is a number from the set $\{0, 1, \ldots, n-1\}$. Shuffling of ballots during elections is implemented through the encryption of identifiers. When encrypting, the following conditions must hold:

1. The encryption result is again an integer from $\{0, 1, \ldots, n-1\}$.
2. The encryption process must involve the block cipher AES with a fixed key K.
3. The number of requests to AES_K must be the same for each identifier.
4. In order to manage security assurances, it should be possible to customize the number of requests to AES_K.

Suggest a way how to organize the required encryption process of identifiers for $n = 5818342$ and $n = 5818343$. In other words, propose a method for organizing a bijective mapping from $\{0, 1, \ldots, n-1\}$ to itself that satisfies conditions described above.

Problem 4. «Let's decode!»

Bob realized a cipher machine for encoding integers from 0 to $n-1$ by 128-bit strings using the secret function Enc. He set $n = 1060105447831$. The cipher machine works as follows: it takes as input a pair of non-negative decimal integers x and d and returns

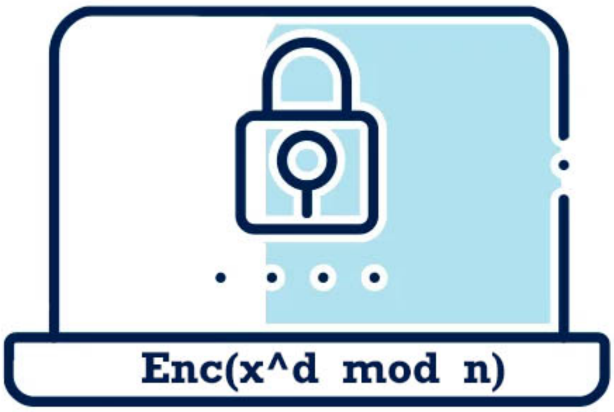
$$\texttt{Enc}(x^d \bmod n).$$

Bob chose a secret number k from 0 to $n-1$ and asked Alice to guess it. Alice said that she can find k if Bob provides her with the cipher machine with an additional property. Namely, x can be also of the form "k", and then the cipher machine will return $\texttt{Enc}(k^d \bmod n)$. In particular, for the query "$k, 1$", the cipher machine returns

$$\texttt{Enc}(k) = \texttt{41b66519cf4356cbbb4e88a4336024da}$$

(the result is in hexadecimal notation). Here it is the cipher machine!

Prove Alice is right and find k with as few requests to the cipher machine as possible!



Problem 5. «Nonlinear hiding»

Nicole is learning about secret sharing. She created a binary vector $y \in \mathbb{F}_2^{6560}$ and splitted it into 20 shares $x_i \in \mathbb{F}_2^{6560}$ (here $\oplus$ denotes the bit-wise XOR):

$$y = x_1 \oplus x_2 \oplus \ldots \oplus x_{20}.$$

Then, she created 20 more random vectors $x_{21}, \ldots, x_{40}$ and shuffled them together with the shares $x_1, \ldots, x_{20}$. Formally, she chose a secret permutation σ of $\{1, \ldots, 40\}$ and computed

$$\begin{aligned} z_1 &= x_{\sigma(1)}, \\ z_2 &= x_{\sigma(2)}, \\ &\ldots \\ z_{40} &= x_{\sigma(40)}, \end{aligned}$$

where each vector $z_i \in \mathbb{F}_2^{6560}$. Finally, she splitted each z_i into 5-bit blocks, and applied a secret bijective mapping $\rho : \mathbb{F}_2^5 \to \mathcal{S}$, where

$$\mathcal{S} = \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}, \mathtt{k}, \mathtt{l}, \mathtt{m}, \mathtt{n}, \mathtt{o}, \mathtt{p}, \mathtt{q}, \mathtt{r}, \mathtt{s}, \mathtt{t}, \mathtt{u}, \mathtt{y}\}$$

(this strange alphabet has `y` instead of `v`).

Formally, she computed $Z_i \in \mathcal{S}^{1312}, 1 \leqslant i \leqslant 40$ such that

$$Z_i = (\rho(z_{i,1\ldots5}), \rho(z_{i,6\ldots10}), \ldots, \rho(z_{i,6556\ldots6560})).$$

After Nicole came back from school, she forgot all the details! She only has written all the Z_i and she also remembers the first 6432 bits of y (128 more are missing). The attachment contains the 6432-bit prefix of y on the first line and $Z_1, \ldots, Z_{40} \in \mathcal{S}^{1312}$ on the following lines, one per line.

Help Nicole to recover full y!

Problem 6. «Studying Feistel schemes»

The classical Feistel scheme and its generalizations are widely used to construct iterated block ciphers. **Generalized Feistel schemes** (GFS) usually divide a message into m subblocks and applies the (classical) Feistel transformation for a fixed number of two subblocks, and then performs a cyclic shift of m subblocks.

Trudy wants to compare algebraic properties of different generalizations of the Feistel scheme based on shift registers over an arbitrary finite commutative ring with identity. For studying, she chooses a nonlinear feedback shift register (NLFSR), Type-II GFS and Target-Heavy (TH) GFS. She wants to decide whether or not these transformations belong to the **alternating group** (that is the group of all even permutations). Trudy needs your help!

Let us give necessary notions. By $A(X)$ we denote the alternating group on a set X. Let t be a positive integer, $t \geqslant 1$, $(R, +, \cdot)$ be a commutative ring with identity 1, $|R| = 2^t$. The characteristic $\text{char}(R)$ of R is equal to 2^c for some $c \in \{1, ..., t\}$. In many block ciphers, we have

$$R \in \left\{\mathbb{Z}_2^t,\ \mathbb{Z}_{2^t},\ \mathbf{GF}(2^t)\right\},\ \text{char}(\mathbb{Z}_{2^t}) = 2^t,\ \text{char}\left(\mathbb{Z}_2^t\right) = \text{char}\left(\mathbf{GF}(2^t)\right) = 2.$$

Q1 NLFSR. Let $\ell \geqslant 1$, $m = 2^\ell$, $h : R^{m-1} \to R$. Consider a mapping $g_{k,h}^{(\text{NLSFR})} : R^m \to R^m$ defined by

$$g_{k,h}^{(\text{NLSFR})} : (\alpha_1, ..., \alpha_m) \mapsto (\alpha_2, \alpha_3, ...,\ \alpha_{m-1}, \alpha_m, \alpha_1 + h(\alpha_2, ..., \alpha_m) + k)$$

for all $(\alpha_1, ..., \alpha_m) \in R^m$, $k \in R$. Describe all positive integers $t \geqslant 1$, $\ell, c \geqslant 1$ and a mapping $h : R^{m-1} \to R$ such that $g_{k,h}^{(\text{NLSFR})} \in A(R^m)$ for any $k \in R$. Prove your answer!

Q2 Type-II GFS. Let $\ell \geqslant 2$, $m = 2^\ell$, $h = (h_1, ..., h_{m/2})$, where $h_i : R \to R$ for $1 \leqslant i \leqslant m/2$. Consider a mapping $g_{k,h}^{(\text{GFS}-\text{II})} : R^m \to R^m$ defined by

$$g_{k,h}^{(\text{GFS}-\text{II})} : (\alpha_1, ..., \alpha_m) \mapsto\ (\alpha_2 + h_1(\alpha_1) + k_1, \alpha_3, \alpha_4 + h_2(\alpha_3) + k_2, \alpha_5, ..., \alpha_{m-1}, \alpha_m + h_{m/2}(\alpha_{m-1}) + k_{m/2}, \alpha_1)$$

for all $(\alpha_1, ..., \alpha_m) \in R^m$, $k = (k_1, ..., k_{m/2}) \in R^{m/2}$. Describe all positive integers $t \geqslant 2$, $\ell, c \geqslant 1$ and mappings $h_1, ..., h_{m/2}$ such that $g_{k,h}^{(\text{GFS}-\text{II})} \in A(R^m)$ for any $k \in R^{m/2}$. Prove your answer!

Q3 TH-GFS Let $\ell \geqslant 2$, $m = 2^\ell$, $h = (h_2, ..., h_m)$, where $h_i : R \to R$ for $2 \leqslant i \leqslant m$. Consider a mapping $g_{k,h}^{(\text{TH})} : R^m \to R^m$ defined by

$$g_{k,h}^{(\text{TH})} : (\alpha_1, ..., \alpha_m) \mapsto (\alpha_2 + h_2(\alpha_1) + k_2, \alpha_3 + h_3(\alpha_1) + k_3, ..., \alpha_{m-1} + h_{m-1}(\alpha_1) + k_{m-1}, \alpha_m + h_m(\alpha_1) + k_m, \alpha_1)$$

for all $k = (k_2, ..., k_m) \in R^{m-1}$. Describe all positive integers $t \geqslant 2$, $\ell, c \geqslant 1$ and mappings $h_2, ..., h_m$ such that $g_{k,h}^{(\text{TH})} \in A(R^m)$ for any $k \in R^{m-1}$. Prove your answer!

Problem 7. «s-Boolean sharing»

In cryptography, a field known as **side-channel analysis** uses extra information such as the power consumption of an implementation to break a cryptographic primitive. In order to defend against these attacks, one does not need to change the primitive but only the way the primitive is implemented. A popular countermeasure is called "**sharing**" where the computation of the primitive is split in multiple parts. Each part seemingly operates on random data such that an adversary has to observe all parts of the computation in order to gain sense of the secret information that was processed.

- An s-**Boolean sharing of a variable** $x \in \mathbb{F}_2$ is a vector $(x_1, x_2, ..., x_s) \in \mathbb{F}_2^s$ such that $x = \bigoplus_{i=1}^s x_i$.
- A vectorial Boolean function $G : \mathbb{F}_2^{sn} \to \mathbb{F}_2^{sm}$ is an s-**Boolean sharing of a function** $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ if for all $x \in \mathbb{F}_2^n$ and $(x_1, ..., x_s) \in \mathbb{F}_2^{sn}$, $x_i \in \mathbb{F}_2^n$, such that $\bigoplus_{i=1}^s x_i = x$,

$$\bigoplus_{i=1}^{s} G_i(x_1, ..., x_s) = F(x)\,.$$

Here, $G = (G_1, ..., G_s)$, where $G_i : \mathbb{F}_2^{sn} \to \mathbb{F}_2^m$ and "$\oplus$" denotes the bit-wise XOR.

Q1 Write an algorithm which takes in a vectorial Boolean function and an integer s and returns true/false on whether the function is a s-Boolean sharing of another function. In case the result is true, the algorithm also returns the function whose sharing is the algorithm's input.

Q2 Problem for a special prize! Propose a theoretical solution to the problem of checking whether the function is a s-Boolean sharing of another function.

Example. If you give the Boolean function $G : \mathbb{F}_2^6 \to \mathbb{F}_2^3$ such that

$$\begin{aligned} G_1(a,b,c,d,e,f) &= ad \oplus ae \oplus bd \\ G_2(a,b,c,d,e,f) &= be \oplus bf \oplus ce \\ G_3(a,b,c,d,e,f) &= cf \oplus cd \oplus af \end{aligned}$$

the algorithm should return true when $s = 3$ together with the function $F : \mathbb{F}_2^2 \to \mathbb{F}_2$ such that $F(x, y) = xy$, where $x = a \oplus b \oplus c$ and $y = d \oplus e \oplus f$.

Problem 8. «Quantum error correction»

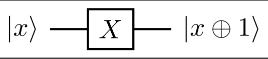
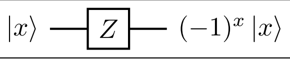
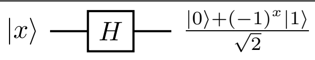
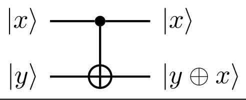
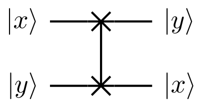
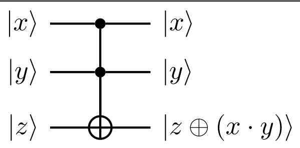
The procedure of error correction is required for quantum computing due to intrinsic errors in quantum gates. One of approaches to quantum error correction is to encode quantum information in three-qubit states, i. e. $\alpha_0 |0\rangle + \alpha_1 |1\rangle \to \alpha_0 |000\rangle + \alpha_1 |111\rangle$. Below are **Problems for a special prize!**

Q1 Design a circuit which implements such encoding.

Q2 Design a circuit which restores the initial state of the three-qubit system, if a single bit-flip error $|0\rangle \leftrightarrow |1\rangle$ occurs in one of three qubits. Hint: use two additional qubits and three-qubit Toffoli gates.

Q3 What will happen, if the quantum gates used for error correction are imperfect? What will be the threshold for gate fidelity, when the error correction will stop working?

Remark. A qubit is a two-level quantum mechanical system whose state $|\psi\rangle$ is the superposition of basis quantum states $|0\rangle$ and $|1\rangle$. The superposition is written as $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$, where α_0 and α_1 are complex numbers that possess $|\alpha_0|^2 + |\alpha_1|^2 = 1$. The amplitudes α_0 and α_1 have the following physical meaning: after the measurement of a qubit which has the state $|\psi\rangle$, it will be found in the state $|0\rangle$ with probability $|\alpha_0|^2$ and in the state $|1\rangle$ with probability $|\alpha_1|^2$. In order to operate with multi-qubit systems, we consider the bilinear operation $\otimes : |x\rangle, |y\rangle \to |x\rangle \otimes |y\rangle$ on $x, y \in \{0, 1\}$ which is defined on pairs $|x\rangle, |y\rangle$, and by bilinearity is expanded on the space of all linear combinations of $|0\rangle$ and $|1\rangle$. When we have two qubits in states $|\psi\rangle$ and $|\varphi\rangle$ correspondingly, the state of the whole system of these two qubits is $|\psi\rangle \otimes |\varphi\rangle$. In general, for two qubits we have $|\psi\rangle = \alpha_{00}|0\rangle \otimes |0\rangle + \alpha_{01} |0\rangle \otimes |1\rangle + \alpha_{10} |1\rangle \otimes |0\rangle + \alpha_{11} |1\rangle \otimes |1\rangle$. The physical meaning of complex numbers α_{ij} is the same as for one qubit, so we have the essential restriction $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$. We use more brief notation $|a\rangle \otimes |b\rangle \equiv |ab\rangle$. For the case of multi-qubit systems with n qubits the general form of the state is $|\psi\rangle = \sum\limits_{(i_1 i_2 \ldots i_n) \in \{0,1\}^n} \alpha_{i_1 i_2 \ldots i_n} |i_1 i_2 \ldots i_n\rangle$.

Pauli-X gate		acts on a single qubit in the state $\lvert x\rangle$, $x \in \{0,1\}$
Pauli-Z gate		acts on a single qubit in the state $\lvert x\rangle$, $x \in \{0,1\}$
Hadamard gate	 $\lvert x\rangle$ H $\frac{\lvert 0\rangle + (-1)^x \lvert 1\rangle}{\sqrt{2}}$	acts on a single qubit in the state $\lvert x\rangle$, $x \in \{0,1\}$
controlled NOT (CNOT) gate	 $\lvert x\rangle$ $\lvert y\rangle$ $\lvert x\rangle$ $\lvert y \oplus x\rangle$	acts on a pair of qubits in the states $\lvert x\rangle, \lvert y\rangle$, $x, y \in \{0,1\}$
SWAP gate	 $\lvert x\rangle$ $\lvert y\rangle$ $\lvert y\rangle$ $\lvert x\rangle$	acts on a pair of qubits in the states $\lvert x\rangle, \lvert y\rangle$, $x, y \in \{0,1\}$
Toffoli gate	 $\lvert x\rangle$ $\lvert y\rangle$ $\lvert z\rangle$ $\lvert x\rangle$ $\lvert y\rangle$ $\lvert z \oplus (x \cdot y)\rangle$	acts on a triple of qubits in the states $\lvert x\rangle, \lvert y\rangle, \lvert z\rangle$, $x, y, z \in \{0,1\}$

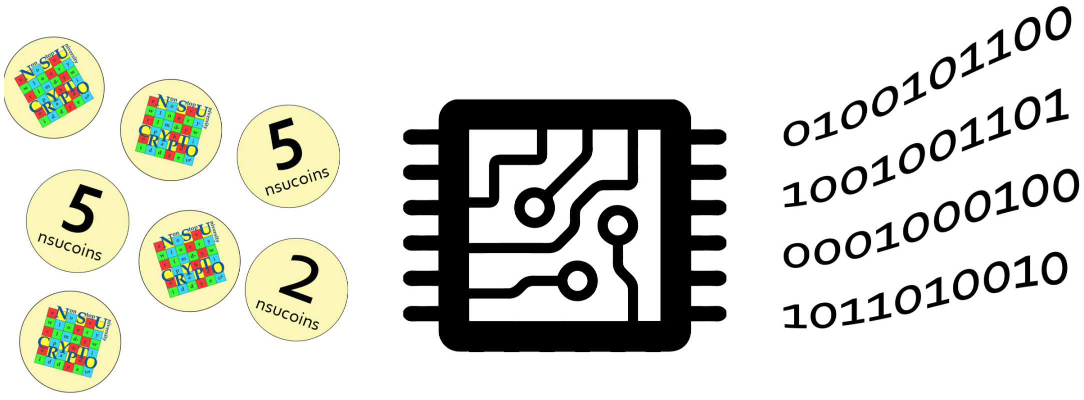
Problem 9. «2021-bit key»

A pseudo-random generator produces sequences of bits (that is of 0 and 1) step by step. To start the generator, one needs to pay 1 *nsucoin* and the generator produces a random bit (that is a sequence of length 1). Then, given a generated sequence S of length ℓ, $\ell \geqslant 1$, one of the following operations can be applied on each step:

1. A random sequence of 4 bits is added to S, so a new sequence S' has length $\ell + 4$. The charge for using this operation is 2 *nsucoins*.

2. A random sequence of 2ℓ bits is added to S, so a new sequence S' has length 3ℓ. The charge for using this operation is 5 *nsucoins*.

Bob needs to generate a secret key of length exactly 2021 bits for his new cipher. What is the minimal number of *nsucoins* that he has to pay for the key?



Problem 10. «Close to permutations»

Bob wants to use a new function inside the round transformation of a cipher. He chooses a family $\mathcal{F}$ of functions F_α from $\mathbb{F}_2^n$ to itself of the form

$$F_\alpha(x) = x \oplus (x \boxplus \alpha), \text{ where}$$

- $x, \alpha \in \mathbb{F}_2^n$,
- $\oplus$ denotes the bit-wise XOR of binary vectors,
- $\boxplus$ denotes the addition modulo 2^n of integers whose binary representations are the given vectors.

Bob noted that functions from $\mathcal{F}$ are not bijective. So, he introduced a parameter that measures in some sense the closeness of a function to a permutation. For a given function F from $\mathbb{F}_2^n$ to itself, the parameter is

$$C(F) = \#\{(x, y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \ : \ F(x) = F(y)\}.$$

The **smaller** the parameter value, the **better** the function. Bob wants to choose «**the best functions**» by this parameter among $\mathcal{F}$. Help Bob to find answers to the questions below!

Q1 How many «the best functions» exist in $\mathcal{F}$?

Q2 What α correspond to «the best functions» from $\mathcal{F}$?

Q3 What is $C(F_\alpha)$ for «the best functions» from $\mathcal{F}$?

Problem 11. «Distance to affine functions»

Given two functions F and G from $\mathbb{F}_2^n$ (or $\mathbb{F}_{2^n}$) to itself, their Hamming distance equals by definition the number of inputs x at which $F(x) \neq G(x)$. The minimum Hamming distance between any such function F and all affine functions A is known to be strictly smaller than $2^n - n - 1$ if $n \geqslant 4$.

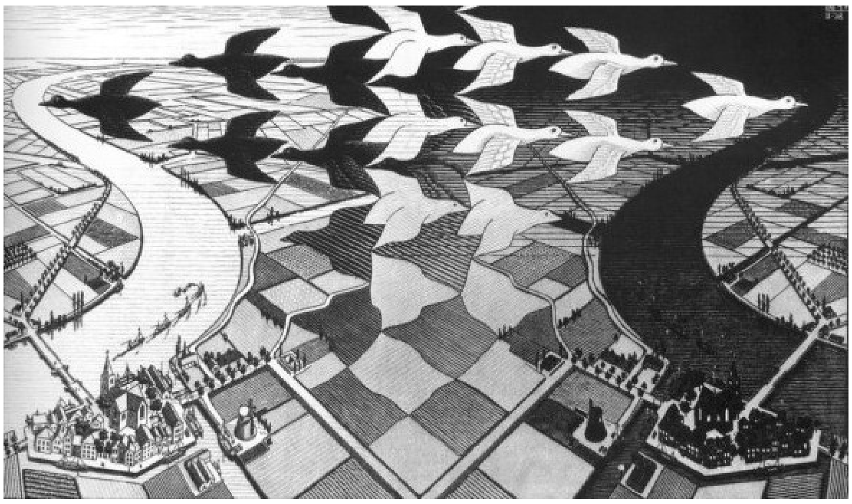
Consider the following problems. Each of them is a **Problem for a special prize!**

Q1 Find a better upper bound valid for every n.

Q2 If **Q1** is unsuccessful, find constructions of infinite classes of functions F having a distance to affine functions as large as possible (infinite classes meaning that these functions are in numbers of variables ranging in an infinite set, such as all positive integers, possibly of some parity for instance).

Q3 If **Q1** and **Q2** are unsuccessful, find constructions (possibly with a computer; then a representation of these functions will be needed, such as their algebraic normal form or their univariate representation) of functions F in fixed numbers of variables having a distance to affine functions as large as possible.

Remark. We recall that an affine function A is a function satisfying $A(x)+A(y)+A(z) = A(x+y+z)$ for all inputs x, y, z.



Problem 12. «The number of rounds»

A famous cryptographer often encrypts his personal data using his favourite block cipher. The block cipher has three variants with $r_1 = 10$, $r_2 = 12$ and $r_3 = 14$ rounds. On this occasion, the cryptographer no longer remembers which of the variants he used.

Fortunately, the cryptographer did ask his students to write down the number of rounds for him. However, in a creative mood, the students decided to encrypt it using a custom cipher E_k with a 4-bit block size. As illustrated in Figure below, round i of their construction XORs the i^{th} nibble k_i of the key $k = k_1 \| k_2 \| \ldots \| k_{r+1}$ with the state and then applies the function S given in Table below. Lacking confidence in their own abilities, the students decided to instantiate the cipher E_k with $r = r_1 \cdot r_2 \cdot r_3 + 1 = 1681$ rounds.

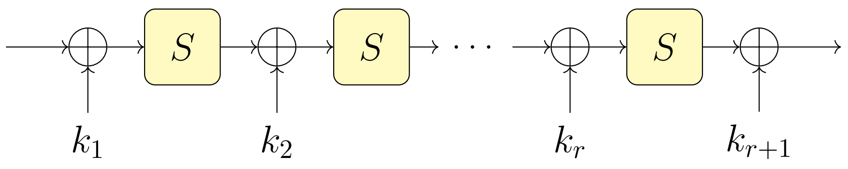


Figure: The students' encryption method.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	3	e	6	8	0	c	b	4	1	d	5	a	7	9	f	2

Table: Lookup table for the function S (in hexadecimal notation).

The students wrote down that the encryption of $r_1 = 10$ is 5 and of $r_2 = 12$ is 0, that is $E_k(1,0,1,0) = (0,1,0,1)$ and $E_k(1,1,0,0) = (0,0,0,0)$. Of course, the students forgot the key, but they still remember that it was an ASCII-encoding of a passphrase consisting only of upper- and lower case English letters. After hearing this, the famous cryptographer exclaims that the students have made a mistake.

How did he know that something was wrong?

Problem 13. «A present for you!»

Alice wants to implement the lightweight block cipher PRESENT on a chip. She starts with the bit permutation that is defined in Table and illustrated in Figure below. Clearly, many lines are intersecting, and this would cause a short circuit if the lines were metal wires. Is it possible to avoid this problem by using several "layers," i.e., parallel planes? That is to draw the lines without intersections on each layer. We assume that

- the work area is a rectangle bounded by the lines where input and output bits are placed and the lines of the outermost connections $P(0) = 0$ and $P(63) = 63$;
- input and output bits are ordered; connections are represented by arbitrary curves;
- color of a line indicates the number of its layer, a line can change color several times;
- the point where a line changes color indicates a connection from one layer to another.

Q1 What is the minimun number of layers required for implementing in this way the PRESENT bit permutation?

Q2 Find a systematic approach how to draw a valid solution for the minimum number of layes found in **Q1** and present the drawing!

For your help (but not necessarily), you can use a specific online tool and download the PRESENT bit permutation as in Figure.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Table: Definition of the bit permutation used in PRESENT.
Bit i is moved to bit position $P(i)$.

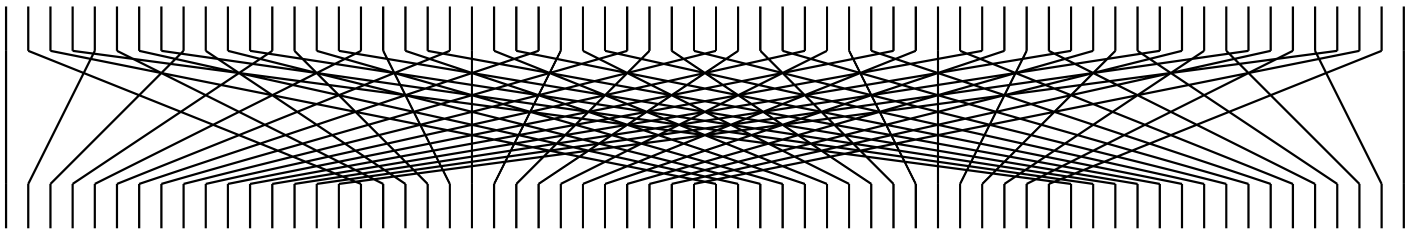


Figure: Illustration of the bit permutation used in PRESENT