Solution of exercise sheet n° 4:

4-1.

(a) The table of the Lempel-Ziv code:

Position	Substring	Output	Output-encoded
1.	A	(0,A)	(00,000)
2.	A_	(1,_)	(01,011)
3.	AB	(1,B)	(01,001)
4.	ABB	(3,B)	(11,001)
5.	ABC	(3,C)	(11,010)
6.	_	(0,_)	(00,011)
7.	ABA	(3,A)	(11,000)
8.	B	(0,B)	(00,001)
9.	C	(0,C)	(00,010)
10.	.	(0,.)	(00,100)

The alphabet of the source reads $\mathcal{A} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \_\}$. To encode the position on can use 2 bits (sufficient to encode 0,1,2,3). Furthermore, one needs three bits to encode the character (taking into account adding the "." at the end). Thus, one needs 5 bits per substring, which gives in total 50 bits. In a naive way one needs $3 \times 18 = 54$ bits to send the sequence. We remark that asymptotically one tends to a number of bits per symbol which is equal to the entropy of the source, and thus smaller than the naive encoding, except in the case of a source with uniform distributed probabilities.

The elements A,B and C of the last ABC appear in different substrings for the first example. For the second example, ABC is part of the sub string that contains the last character.

(b)

Position	Substring	Output	Output-encoded
1.	A	(0,A)	(000,00)
2.	B	(0,B)	(000,01)
3.	AA	(1,A)	(001,00)
4.	AAA	(3,A)	(011,00)
5.	AAAA	(4,A)	(100,00)
6.	AAAAA	(5,A)	(101,00)
7.	BB	(2,B)	(010,01)
8.	.	(0,.)	(000,11)

The alphabet of the source reads $\mathcal{A} = \{\mathtt{A}, \mathtt{B}\}$.
Lempel-Ziv: $5 * 8 = 40$ bits
Naive encoding: $2 * 19 = 38$ bits

(c)

Position	Substring	Output	Output-encoded
1.	A	(0,A)	(000,00)
2.	B	(0,B)	(000,01)
3.	AA	(1,A)	(001,00)
4.	AAA	(3,A)	(011,00)
5.	AAAA	(4,A)	(100,00)
6.	AAAAA	(5,A)	(101,00)
7.	AAAAAA	(6,A)	(110,00)
8.	.	(0,.)	(000,11)

The alphabet of the source reads $\mathcal{A} = \{\mathtt{A}, \mathtt{B}\}$.
Lempel-Ziv: $5 * 8 = 40$ bits
Naive encoding: $2 * 23 = 46$ bits
(d) The original sequence is: AABABC_ ABBBBBBBB.

4-2.

(a) The optimal method of asking questions can be found by considering the Huffman code applied to the source $X_1X_2...X_n$.

In order to apply this code one needs to define a new random variable Y which has an alphabet that contains all outcomes of the sequence $X_1X_2...X_n$ That is,

$$\begin{aligned}\mathcal{Y} = \{ & \underbrace{111...111}_{\text{“All objects are faulty”}}, \underbrace{111...110}_{\text{“The first } n-1 \text{ objects are faulty, the last one is not”}}, \\ & \underbrace{111...100}_{\text{“The first } n-2 \text{ objects are faulty, the last two are not”}}, \\ & \underbrace{111...101}_{\text{“The first } n-2 \text{ objects and the last one are faulty, the } (n-1)\text{th object is not faulty”}}, \\ & ..., \underbrace{000...000}_{\text{“No object is faulty.”}} \}.\end{aligned} \tag{1}$$

In total there are 2^n possible sequences and we allocate to them the following probability distribution

$$\begin{aligned}q(y) = \{ & p_1p_2p_3 \cdots p_n, p_1p_2p_3 \cdots p_{n-2}p_{n-1}(1-p_n), \\ & p_1p_2p_3 \cdots p_{n-2}(1-p_{n-1})(1-p_n), \\ & p_1p_2p_3 \cdots p_{n-2}(1-p_{n-1})p_n, \\ & ..., (1-p_1)(1-p_2) \cdots (1-p_n)\}.\end{aligned} \tag{2}$$

For given numbers for p_1, p_2 etc. we can rank all values of Y according to the probabilities $q(y)$ ($q(y)$ has to be normalized) and then construct the Huffman code.

(b) The longest sequence of questions is associated to the case which is the least probable. Since $p_i > \frac{1}{2}$ the least probable case is the case when all objects are faulty. The second least probable is the case where all except the last one is faulty. These two cases are the leafs of the longest branches of a binary tree associated to the Huffman code. The last question is thus: "The last object, is it faulty?".

4-3.

The player B uses the optimal code. We also know that this player needs to ask, on average, 35 questions ("bits") to identify the object. This implies the inequality

$$\begin{aligned} H(X) &\leq 35 < H(X) + 1 \\ &\Rightarrow 34 < H(X) \end{aligned}$$

For a certain fixed number of objects, the distribution of objects of player A which gives the highest entropy is the uniform distribution. Conversely, for a fixed value of the entropy the uniform distribution minimizes the number of objects that are associated to the fixed entropy. The uniform distribution offers a possibility to calculate a lower bound on the number of objects:

$$34 < \log_2 m.$$

We conclude that there are at least $2^{34} \simeq 1.7\ 10^{10}$ objects in the ensemble.

4-4.

(a) The probability distribution that maximizes the Shannon entropy is the one where all possible elements have the same probability to occur. There are $n+1$ possible elements: n where one of the coins is counterfeited and one element taking into account the possibility that no coin is counterfeited. The probability distribution of this case is $p_i = \frac{1}{n+1}$, and the Shannon entropy is $H(X) = \log_2(n+1)$ bits.

(b) With the optimal code, one needs on average k weighs with k such that

$$H(X) \leq k < H(X) + 1. \tag{3}$$

For the case of maximal entropy, we have

$$\log_2(n+1) \leq k < \log_2(n+1) + 1. \tag{4}$$

(c) With the help of (b) one deduces that $n + 1 \leq 2^k < 2(n+1)$, thus, $n \leq 2^k - 1$.

(d) The probability distribution attains the bound, if $n = 2^k - 1$ and thus $p_i = \frac{1}{2^k}$ (probability distribution which is 2-adic).

Weigh	Max. coins
1	1
2	3
3	7
4	15

The bound is attained if one has a number of coins equal to $1, 3, 7, 15, \ldots$.

4-5.

(a) We are given $L - H_5(X) = 0$

$$L - H_5(X) = \sum_{i=1}^{m} p_i l_i + \sum_{i=1}^{m} p_i \log_5 p_i = 0,$$

but $l_i = -\log_5 5^{-l_i}$, so we can rewrite

$$L - H_5(X) = -\sum_{i=1}^{m} p_i \log_5 5^{-l_i} + \sum_{i=1}^{m} p_i \log_5 p_i = 0.$$

Defining $r_i = \frac{5^{-l_i}}{\sum_j 5^{-l_j}}$ and $R = \sum_i 5^{-l_i}$, we have

$$L - H_5(X) = \sum_i p_i \log_5 \frac{p_i}{r_i} - \log_5 R = 0.$$

The Kraft inequality tells us that $R \leq 1$ and we remark that r_i is a probability distrubtion, since $r_i \geq 0$ and $\sum_i r_i = 1$, so we can rewrite the previous expression, using the definition of the relative entropy $D(x||y)$:

$$L - H_5(X) = D(p||r) + \log_5 \frac{1}{R} = 0.$$

We know that $D(p||r) \geq 0$ and $\log_5 \frac{1}{R} \geq 0$ since $R \leq 1$. The inequality implies $R = 1$ and $p_i = r_i = 5^{-l_i}$.

(b) Since $L = H_5(X)$ we know that the 5-adic code is optimal. Thus, we can construct it using the Huffman code. For each step of this encoding one regroups 5 elements to one, thus the number of elements is decreased by 4 at each step. If at the beginning one had m elements, at the end of the encoding, i.e. after k steps, there remains only one element $(m - 4k = 1)$. Thus, $m = 4k + 1$.