Exercise 1

(a)

$$C_1 = \{00000, 11100, 01110, 00111, 10011, 11001\}$$

No, it's not a cyclic code, because it's not a linear code. $11100 + 01110 = 10110 \notin C_1$

(b) $C_2 = C \cup (1\dots1 + C)$, where C is binary cyclic code and $1\dots1$ is all 1 codeword.
Yes, because it is linear code:
for any $c_1, c_2 \in C$ and $c_1', c_2' \in (1\dots1 + C)$, where $c_1' = c_1 + 1\dots1$ and $c_2' = c_2 + 1\dots1$

- $c_1' + c_2' = (c_1 + 1\dots1) + (c_2 + 1\dots1) = c_1 + c_2 + 2 \cdot 1\dots1 = c_1 + c_2$ and $(c_1 + c_2) \in C$
- $c_1' + c_2 = (c_1 + 1\dots1) + c_2 = (c_1 + c_2) + 1\dots1$ and $(c_1 + c_2) \in C$ and therefore $(c_1 + c_2 + 1\dots1) \in (C + 1\dots1)$
- $c_1 + c_2 \in C$

and because operation $+1\dots1$ just flips all bits independently on their position, following holds: $\sigma(c) + 1\dots1 = \sigma(c + 1\dots1)$
and therefore if $c, \sigma(c) \in C$ then $(c + 1\dots1), \sigma(c + 1\dots1) \in (C + 1\dots1)$

Exercise 2

Let σ denote a *circular right shift* operation. Find all binary words w

(a) of length n such that $\sigma(w) = w$;
for $w = w_1 \dots w_n$, $w_n w_1 \dots w_{n-1} = w_1 w_2 \dots w_n$
therefore $w_i = w_{i-1 \bmod n}$ and that holds only for 0^n and 1^n

(b) of length 6 such that $\sigma^2(w) = w$;
for $w = w_1 \dots w_n$, $w_{n-1} w_n w_1 \dots w_{n-2} = w_1 w_2 w_3 \dots w_n$
therefore $w_i = w_{i-2 \bmod n}$ and that holds only for $000000, 111111, 010101, 101010$

(c) of length 6 such that $\sigma^3(w) = w$. for $w = w_1 \dots w_n$, $w_i = w_{i-3 \bmod n}$ must hold and that holds only for $000000, 001001, 010010, 011011, 100100, 101101, 110110, 111111$

Exercise 3

1. (a) This is a straightforward use of the definition from the slides. Codes from R_7 have length and since $deg(1+x^2+x^3)=3$, the dimension of the code is $n-3=4$. We need to write four cyclic shifts of 1011000 as rows of the matrix:

$$G=\begin{pmatrix}1&0&1&1&0&0&0\\0&1&0&1&1&0&0\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{pmatrix}$$

   (b) According to slides $x^7-1=(1+x^2+x^3)\cdot h(x)$ and the dual code of C is generated by reciprocal $\bar{h}(x)$ of $h(x)$. We thus need to find $h(x)$ first. This is done by polynomial division. We get that $h(x)=1+x^2+x^3+x^4$ and thus $\bar{h}(x)=1+x+x^2+x^4$. Since dual of C has dimension 3, three cyclic shifts of 1110100 are rows of H. Thus:

$$H=\begin{pmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\0&0&1&1&1&0&1\end{pmatrix}$$

   (c) In order to encode with the use of polynomials we need to make an identity between the message $m=1010$ and a polynomial message $m(x)=1+x^2$. Now

$$\begin{aligned}c(x)&=m(x)g(x)\mod x^7-1\\&=(1+x^2)(1+x^2+x^3)\\&=1+x^2+x^3+x^2+x^4+x^5\\&=1+x^3+x^4+x^5.\end{aligned}$$

Exercise 4

3.4 (a) How many binary cyclic codes of length 9 are there?

Solution: We are in $\mathbb{F}_2$. First of all we have to factorize $x^9 - 1 = x^9 + 1$ into a irreducible polynomials over $\mathbb{F}_2$.

$$x^9 + 1 = (x+1)(x^2 - x + 1)(x^6 - x^3 + 1)$$

All binary cyclic codes are in form:

$$(x+1)^{a_1}(x^2 - x + 1)^{a_2}(x^6 - x^3 + 1)^{a_3}; \forall i \in \{1,2,3\}, a_i \in \{0,1\}$$

We have 2^3 possibilities, therefore there are **8** binary cyclic codes of length 9.

□

(b) How many ternary cyclic codes of length 9 are there?

Solution: We are in $\mathbb{F}_3$. First of all we have to factorize $x^9 - 1 = x^9 + 2$ into a irreducible polynomials over $\mathbb{F}_3$.

$$x^9 + 2 = (x+2)^9$$

All binary cyclic codes are in form:

$$(x+2)^{a_1}, a_1 \in \{0,9\}$$

We have 10 possibilities, therefore there are **10** ternary cyclic codes of length 9. □

(c) How many ternary cyclic codes of length 9 have dimension 7?

Solution: We have to factorize $x^9 - 1$ over $\mathbb{F}_3$. As above we have $x^9 + 2 = (x+2)^9$. We have $[9,7]$-code, therefore we have to find factors of degree:

$$deg(f(x)) = n - k = 9 - 7 = 2.$$

Therefore we have only **1** code of degree 2 with generator polynomial $(x+2)^2$. □

Exercise 5

Let C_1 and C_2 be cyclic codes with generator polynomials $g_1(x)$ and $g_2(x)$. Then $C_1 \subseteq C_2$ iff $g_2(x)$ divides $g_1(x)$

Since we know from slides that $C^\perp$ is generated by $\bar{h}(x)$, we obtain the proof by setting $C_1 = C$ and $C_2 = C^\perp$ in the above theorem. The nice proof is as follows. Essentially we want to prove the following:

$$\langle g_1 \rangle \subseteq \langle g_2 \rangle \Leftrightarrow g_2 | g_1$$

"$\Leftarrow$" If $g_2|g_1$, there exists h, such that $g_1 = hg_2$. Since by definition $C_2 = \langle g_2 \rangle = \{f(x)g_2(x) \mod x^n - 1 | f(x) \in R_n\}$, it in particular contains all elements in $\langle hg_2 \rangle = \{f(x)h(x)g_2(x) \mod x^n - 1 | f(x) \in R_n\} = C_1$. "$\Rightarrow$" Since $\langle g_1 \rangle \subseteq \langle g_2 \rangle$, for each $f(x) \in R_n$ there exists $h(x)$, such that $fg_1 = hg_2$. In particular this holds for $f(x) = 1$, meaning there exists $h(x)$ such that $g_1 = hg_2$, in other words $g_2|g_1$.

Proof. C is self-orthogonal if and only if the reciprocal polynomial $\bar{h}(x)$ divides $g(x)$.

we know that $C \subseteq C' \iff g(x)$ is divisible by $g'(x)$

and we know that generator polynomial of $C^{\perp}$ is $\bar{h}(x)$

therefore we know that $C \subseteq C^{\perp} \iff g(x)$ is divisible by $g^{\perp}(x)$

Exercise 6

Let $g(x) = gcd(g_1(x), g_2(x))$, where gcd is the polynomial greatest (highest degree) common divisor of the two polynomials and let C' be the code generated by $g(x)$.

First we show that a code generated by $g(x)$ contains $C_1 \cup C_2$: if a codeword is in this union then clearly it is a multiple of $g_1(x)$ or $g_2(x)$. Now since $g(x)$ is a divisor of both $g_1(x)$ and $g_2(x)$ the codeword is also a multiple of $g(x)$ and thus is in a code generated by $g(x)$.

Now we know by definition of gcd (using the extended euclidean algorithm) that

$$g(x) = a(x)g_1(x) + b(x)g_2(x) \mod (x^n - 1),$$

for some polynomials $a(x), b(x)$ and where n is the code block length. Now $a(x)g_1(x) \in C_1$ and $b(x)g_2(x) \in C_2$. From this we can see that $g(x)$ is just a sum of two codewords from C_1 and C_2 respectively. Any cyclic (and thus linear) code containing $C_1 \cup C_2$ (and thus $a(x)g_1(x)$ and $b(x)g_2(x)$) must also contain this sum of the two code words and thus must contain $g(x)$ and all its multiples, thus $C' \subseteq C$ and with the previous result we obtain $C = C'$ and $g(x)$ is our desired polynomial. (The polynomial gcd is unique up to a multiplication by a constant but this constant doesn't change the code C'.)

Exercise 7

(a) $RM(1,3) =$
{00000000, 00001111, 00110011, 00111100, 01010101, 01011010, 01100110, 01101001, 10010110, 10011001, 10100101, 10101010, 11000011, 11001100, 11110000, 11111111 }

$RM(1,4) =$
$\{$0000000000000000, 0000000011111111, 0000111100001111, 0000111111110000, 0011001100110011, 0011001111001100, 0011110000111100, 0011110011000011, 0101010101010101, 0101010110101010, 0101101001011010, 0101101010100101, 0110011001100110, 0110011010011001, 0110100101101001, 0110100110010110, 1001011001101001, 1001011010010110, 1001100101100110, 1001100110011001, 1010010101011010, 1010010110100101, 1010101001010101, 1010101010101010, 1100001100111100, 1100001111000011, 1100110000110011, 1100110011001100, 1111000000001111, 1111000011110000, 1111111100000000, 1111111111111111 $\}$

(b) From the construction, we can see that if we find a base generating the (u, u) part of the code, then we can get $(u, u + \overrightarrow{1})$ with the help of one vector that is $(\overrightarrow{0}, \overrightarrow{1})$, where $|\overrightarrow{0}| = |\overrightarrow{1}|$. So all we need to solve is the remaining vectors. The idea of the code is repetition, so the new base words will be repetitions of all the old, this gives us the construct

$$G(1,0) = [1] \text{ and } G(1,n) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } G(1,m) = \begin{bmatrix} G(1,m-1) & G(1,m-1) \\ \overrightarrow{0} & \overrightarrow{1} \end{bmatrix}$$

(c) We can show that the parameter with an induction like proof on m.

Base case: $R(1,1)[2,2,1]$ is clearly an $[2^m, m+1, 2^{m-1}]$ code.

Induction hypothesis: We assume that $R(1, m-1)$ is a $[2^{m-1}, m, 2^{m-2}]$ code.

Induction Step: The repetition always doubles the length so $n = 2 \times 2^{m-1} = 2^m$.
Also we see that repetition itself does not add a new base word, only the "reversed" repetition needs to be handled, and that is done by one word $(\overrightarrow{0}, \overrightarrow{1})$ so $k = m + 1$
For d, we need to check two options:
1. By only repeating, the smallest distance(weight) will double, what gives

$$2 \times 2^{m-2} = 2^{m-1}$$

2. if we do the "reversed" repetition, (and the base contains x 1-bits) we are adding $2^{m-1} - x$ to the weight, which in total results in

$$2^{m-1} - x + x = 2^{m-1}$$

.

And d is the minimum of the two, which is 2^{m-1}.

For the weights, if for the simple repetition case $v = (u, u)$, and u is $\overrightarrow{0}$ or $\overrightarrow{1}$, than v will also be $\overrightarrow{0}$ or $\overrightarrow{1}$. Every other word will have $2 \times 2^{m-2} = 2^{m-1}$ weight(as seen in the induction above). For the case $v = (u, u + \overrightarrow{1})$ there are 3 cases:
1. $u = \overrightarrow{0}$, then we get $(\overrightarrow{0}, \overrightarrow{1})$ where clearly half of the bits is 1, what gives weight

$$2^m \div 2 = 2^{m-1}$$

2. $u = \overrightarrow{1}$, the case is the same as in point 1 above, just with $(\overrightarrow{1}, \overrightarrow{0})$.
3. For every other u, as showed in the induction step above(point 2), the weight will be half of the length, which is 2^{m-1}.