Question 1.

(a) $C = \{11111, 00100, 10010, 01001\}$

$h(11111, 00100) = 4$
$h(11111, 10010) = 3$
$h(11111, 01001) = 3$
$h(00100, 10010) = 3$
$h(00100, 01001) = 3$
$h(10010, 01001) = 4$

$h(C) = h(11111, 10010) = 3.$

(b) Strings 11011, 01101, 10011, 00111 decoded according to the nearest neighbor strategy

$h(11011, 11111) = 1$
$h(11011, 00100) = 5$
$h(11011, 10010) = 2$
$h(11011, 01001) = 2$
11011 $\mapsto$ 11111

$h(01101, 11111) = 2$
$h(01101, 00100) = 2$
$h(01101, 10010) = 5$
$h(01101, 01001) = 1$
01101 $\mapsto$ 01001

$h(10011, 11111) = 2$
$h(10011, 00100) = 3$
$h(10011, 10010) = 1$
$h(10011, 01001) = 3$
10011 $\mapsto$ 10010

$h(00111, 11111) = 2$
$h(00111, 00100) = 2$
$h(00111, 10010) = 3$
$h(00111, 01001) = 3$
00111 $\mapsto$ 11111 or 00100

Question 2.

(a) The number of codewords is set to 4.

We will start with $N = 1$, for which we create $C_{base} = \{00, 01, 10, 11\}$ which is a $(2N, 4, N)$ code according to N.

We will construct the code for any N as follows – each codeword will be constructed as a codeword from C_{base} concatenated N-times.

So since C_{base} is a $(2, 4, 1)$ code, for any N the codewords will be of length $2N$.
And since the first codeword c_1 will always consist only of 0s, c_2 and c_3 will have N number of 1s on mutually opposite positions, and c_4 will be only 1s, together it makes the minimum distance $= N$ (the same principle as in C_{base}, where $c_1 = 00$ and $c_2 = 01$ and therefore $d = 1$).

Example:
For $N = 3$ we will get $C = \{000000, 010101, 101010, 111111\}$, which is a $(6, 4, 3)$ code.

(b) We will create the code by creating a base of the code which will be then concatenated d-times. The process is as follows:

- We create M codewords as increasing binary numbers 0, ..., $M - 1$ on $\lceil \log_2 M \rceil$ bits, which is the least amount of bits needed to store value $M - 1$. Since each codeword is a different binary represented number, the minimum distance is at this point 1.
- We get the result codewords by d-times concatenating corresponding base codeword.
  $c_i = (Base_i)^d$, therefore $n = d \cdot \lceil \log_2 M \rceil$
  Since the minimum distance of base code is 1, the minimum distance of C is d.

Example:
$M = 5, d = 3$. We must construct $(n, 5, 3)$ code.
The base codewords will be of length $\lceil \log_2 5 \rceil = 3$ bits.
$Base = \{000, 001, 010, 011, 100\}$
Concatenate each base codeword d-times:
$C = \{000000000, 001001001, 010010010, 011011011, 100100100\}$

Question 3.

(a) After increasing the middle bit in code words of code $C2$ by 1 (so that 0 becomes 1, 1 becomes 2 and 2 becomes 0), we get code with code words $\{000, 222, 111\}$.

Codes $C1$ and $C2$ are equivalent.

(b) When we calculate the distances between code words of both codes, we find out that code $C1$ contains a pair of code words with distance 4, while there is no such pair in code $C2$.

(i) Distance of code words in code $C1$:
d(0011, 1010) = 2
d(0011, 0101) = 2
d(1010, 0101) = 4

(ii) Distance of code words in code $C2$:
d(0000, 0110) = 2
d(0000, 0011) = 2
d(0110, 0011) = 2

Therefore, codes $C1$ and $C2$ are not equivalent.

Question 4.

We can map each string from binary code C with (n, M, d) to a vector R^n. Each character c of codeword s_x on position i maps to a number $u_x[i] = l$ if $c = 1$ or $u_x[i] = -l$ if $c = 0$. To create normalized vectors, we will set $l = 1/\sqrt{n}$.
Observe that we can deduce how many characters two codewords s_x and s_y share, by performing dot product of corresponding vectors u_x and u_y: $((u_x \cdot u_y) + 1) \cdot (n/2)$ (if strings share no characters, dot product is -1 and each character they share adds $2/n$ to the dot product).
Proof by contradiction: Let us assume that we have found a binary code C with (n, M, d) where $M > 2n$ and $d \geq n/2$. $C = \{s_1, s_2, \ldots, s_M\}$ and corresponding vectors are $u_1, u_2, \ldots u_M$. Since $d \geq n/2$, dot product of any two different vectors u_x and u_y must be $u_x \cdot u_y \leq 0$ (since s_x and s_y share at most half the characters). Notice, that vectors $u_1, u_2, \ldots, u_M$ now have all the properties that are required by given lemma. But this contradicts given lemma that says that we can find at most $2n$ of vectors with these properties (we have $M > 2n$ of them).

Question 5.

(a) By multiplying digits on even positions by 2, we get 29058x. By summing the digits we get $24 + x$. If we want $24 + x$ to be divisible by 10, then $x = 6$.

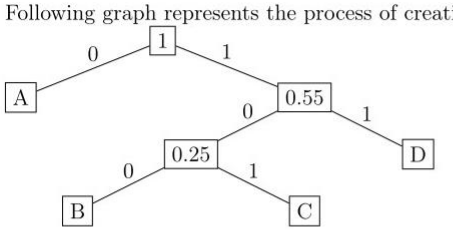
(b) Given digits d_1, d_2, d_3, d_4, d_5 of codeword $d_1d_2d_3d_4d_5x$, we multiply the first, third and fifth digit by 2 (because they're on even positions going from right) and we get $2d_1d_22d_3d_42d_5$. Then get sum S of the digits of $2d_1d_22d_3d_42d_5$ (if $2d_i > 9$ we take its individual digits) and the check digit $x = (10 - (S \mod 10)) \mod 10$

(c) It can detect any single error. If it couldn't, there would a changed digit that was differed from the original one by 10, which is impossible, since digits are 0-9, or a digit that when multiplied by 2 and the sum of the digits of the result differed by 10 from the sum of digits of the original one multiplied by 2. That is also impossible, since the sum of digits of a any digit multiplied by 2 is at most 9.

(d) It can't detect transposition error involving adjacent digits if the digits are 0 and 9, because $2 \cdot 0 + 9 = 0 + 1 + 8$, where $1 + 8$ is the sum of digits of $2 \cdot 9$, meaning if adjacent digits 9 and 0 exchange positions it will still be a valid codeword. Example might be the codewords 190546 and 109546, which both have the same sum of individual (unchanged and doubled) digits.

Question 6.

Following graph represents the process of creation of the Huffman code.



Huffman code C is therefore defined by the following table:

symbol	codeword
A	0
B	100
C	101
D	11

$AVG(C) = \sum_{c \in C} P(c) \cdot |c| = 0.45 \cdot 1 + 0.10 \cdot 3 + 0.15 \cdot 3 + 0.3 \cdot 2 = 1.8$

Average codeword length is 1.8.

$S(X) = -\sum_{x \in X} P(x) \cdot \log_2(P(x))$
$= -1 \cdot (0.45 \cdot \log_2(0.45) + 0.10 \cdot \log_2(0.10) + 0.15 \cdot \log_2(0.15) + 0.30 \cdot \log_2(0.30)) \approx 1.78223$

Efficiency is equal to $\frac{S(X)}{AVG(C)} \approx \frac{1.78223}{1.8} \approx 0.99013$.