Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»
КАФЕДРА ИУК2 «Информационные системы и сети»

ЛАБОРАТОРНАЯ РАБОТА №3

«Машинные коды»

ДИСЦИПЛИНА: «Теоретическая информатика»

Выполнил: студент гр. ИУК4-11Б (Суриков Н.С)
(подпись) (Ф.И.О.)
Проверил: (Гладских А.П)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):
- Балльная оценка:
- Оценка:

Калуга, 2023

Цель работы: сформировать навыки анализа монотонных кодов и методы программирования интервального кодирования. Приобрести навыки сравнения способов кодирования в системах с возможностью помехоустойчивого кодирования.

Задачи:

1. Пользуясь основными правилами и формулами прямого, дополнительного и обратного машинных кодов представить несколько примеров двоичных чисел в этих кодах.
2. Привести примеры целых и дробных, а также положительных и отрицательных чисел.
3. Пользуясь правилами сложения двоичных чисел в машинных кодах, привести несколько примеров на сложение. В примерах должны быть показаны случаи сложения в каждом коде: положительных чисел с положительными, положительных чисел с отрицательными с получением как положительной, так и отрицательной суммы (результата), отрицательных чисел с отрицательными.
4. Составить отчет о проделанной работе. Форма отчета – отчет в электронном виде. Отчет должен содержать: постановку задачи, краткое описание приемов перевода двоичных чисел из естественной формы в прямой, дополнительный и обратный коды и правил сложения чисел в машинных кодах. Примеры перевода и сложения, выполненные по заданию лабораторной работы.

ВАРИАНТ 21

Задание №1

а)16 б)256 в)4096 г)65536

Задание №2

#21: 0x27 0x41 0x6E 0x64 0x20 0x74 0x68 0x6F 0x75 0x20 0x68 0x61 0x730x74 0x20 0x73 0x6C 0x61 0x69 0x6E 0x20 0x74 0x68 0x65 0x20 0x4A 0x61 0x62 0x62 0x65 0x72 0x77 0x6F 0x63 0x6B 0x3F

'And thou hast slain the Jabberwock?

```
s = """0x27 0x41 0x6E 0x64 0x20 0x74 0x68 0x6F 0x75 0x20 0x68 0x61 0x73 0x74
0x20 0x73 0x6C 0x61 0x69 0x6E 0x20 0x74 0x68 0x65 0x20 0x4A 0x61 0x62 0x62
0x65 0x72 0x77 0x6F 0x63 0x6B 0x3F"""

for i in s.split():
    print(chr(int(i, 16)), end='')
```

Задание №3

21. Дизассемблер

0x414 0x438 0x437 0x430 0x441 0x441 0x435 0x43c 0x431 0x43b 0x435 0x440

```
for i in "Дизассемблер":
    print(hex(ord(i)), end=' ')
```

Задание №4

20-12-15-21-13-41-31-20-16-20-41-25-6-13-16-3-6-12-41-19-16-2-1-25-12-21-41-19-3-16-32-41-3-41-19-13-6-5-29-41-3-16-9-13-6-41-5-3-6-18-10-36-41-19-12-1-9-1-13-41-«-17-19-»-41-10-41-16-20-16-26-7-13-35-41-17-16-15-32-23-1-13-1-41-19-16-2-1-12-1-41-3-16-9-5-21-23-36-41-17-16-3-6-13-1-41-17-16-41-20-16-13-17-6-41-4-13-1-9-16-14-41-(-15-1-18-16-5-36-41-12-16-15-6-25-15-16-36-41-19-16-2-18-1-13-19-33-)-41-10-41-3-5-18-21-4-41-12-41-2-1-2-12-6-41-22-7-12-13-6-36-41-19-41-17-33-20-16-4-16-41-15-16-14-6-18-1-36-41-17-16-5-23-16-5-10-20-41-10-41-15-32-23-1-6-20-41-6-11-41-17-16-5-16-13-35-41-2-1-2-12-1-41-9-1-41-20-16-13-17-21-35-41-19-16-2-1-12-1-41-9-1-41-32-2-12-21-35-41-2-1-2-12-1-41-3-41-19-20-16-18-16-15-21-41-34-41-10-41-19-16-2-1-12-1-41-9-1-41-15-6-11-35-41-21-23-3-1-20-10-13-1-41-2-1-2-12-21-41-9-1-41-32-2-12-21-41-10-41-15-6-41-17-21-27-1-6-20-35-
-18-21-23-15-21-13-1-41-2-1-2-12-1-41-15-1-41-12-16-13-6-15-10-41-17-6-18-6-5-41-1-4-6-15-20-16-14-35-
-34-41-5-1-36-34-41-4-16-3-16-18-10-20-36-34-41-17-16-17-1-13-1-19-30-35-41-15-6-41-16-20-17-10-18-1-32-19-30-35-41-10-36-34-41-4-16-3-16-18-10-20-36-34-41-17-33-20-30-41-3-6-5-7-18-41-9-1-12-3-1-19-12-10-41-34-41-31-20-16-41-20-1-12-35-41-10-41-1-17-17-1-18-1-20-41-34-41-31-20-16-41-5-6-11-19-20-3-10-20-6-13-30-15-16-41-3-6-18-15-16-35-41-3-19-7-36-34-41-4-16-3-16-18-10-

20-36-34-41-15-1-23-16-5-10-20-19-33-41-3-41-3-1-15-15-16-11-41-12-16-14-15-1-20-6-35-41-3-6-5-10-20-6-41-14-6-15-33-41-3-41-14-10-13-10-24-10-32-35--15-21-36-41-15-1-18-16-5-36-41-12-16-15-6-25-15-16-36-41-1-23-15-21-13-35--34-41-1-41-26-21-2-1-39-41-34-41-19-17-18-1-26-10-3-1-32-20-35--34-41-17-18-16-41-26-21-2-21-36-34-41-4-16-3-16-18-10-20-36-34-41-15-10-25-6-4-16-41-15-6-41-9-15-1-32-41-10-41-3-6-5-1-20-30-41-15-6-41-3-6-5-1-32-36-41-1-41-16-19-20-1-13-30-15-16-6-41-34-41-31-20-16-41-20-1-12-35-41-3-6-5-10-20-6-41-14-6-15-33-36-41-12-1-9-15-10-20-6-35--15-21-36-41-21-3-6-13-10-41-2-1-2-12-21-35--19-15-16-3-1-41-3-9-33-13-41-1-4-6-15-20-41-19-16-2-1-25-10-27-21-41-19-3-16-32-36-41-19-15-16-3-1-41-20-12-15-21-13-41-6-7-41-15-16-19-16-14-41-3-41-19-13-6-5-29-36-41-19-12-1-9-1-13-41-«-17-19-»-41-10-41-16-20-16-26-7-13-35--17-16-3-6-13-1-41-19-16-2-1-25-10-27-1-41-4-13-1-9-16-14-36-41-17-16-15-32-23-1-13-1-41-17-21-19-20-16-11-41-3-16-9-5-21-23-41-10-41-3-5-18-21-4-41-12-41-4-18-1-8-5-1-15-10-15-21-41-21-17-18-1-3-5-16-14-21-41-17-16-5-23-16-5-10-20-35

ткнул этот человек собачку свою в следы возле двери, сказал «пс» и отошёл. понюхала собака воздух, повела по толпе глазом (народ, конечно, собрался) и вдруг к бабке фёкле, с пятого номера, подходит и нюхает ей подол. бабка за толпу. собака за юбку. бабка в сторону — и собака за ней. ухватила бабку за юбку и не пущает.

рухнула бабка на колени перед агентом.

— да,— говорит,— попалась. не отпираюсь. и,— говорит,— пять ведёр закваски — это так. и аппарат — это действительно верно. всё,— говорит,— находится в ванной комнате. ведите меня в милицию.

ну, народ, конечно, ахнул.

— а шуба? — спрашивают.

— про шубу,— говорит,— ничего не знаю и ведать не ведаю, а остальное — это так. ведите меня, казните.

ну, увели бабку.

снова взял агент собачищу свою, снова ткнул её носом в следы, сказал «пс» и отошёл.

повела собачища глазом, понюхала пустой воздух и вдруг к гражданину управдому подходит.

Задание №5

24-16-19-25-17-4-35-24-20-24-4-29-10-17-20-7-10-16-4-23-20-6-5-29-16-25-4-23-7-20-36-4-7-4-23-17-10-9-33-4-7-20-13-17-10-4-9-7-10-22-14-40-4-23-16-5-13-5-17-4-«-21-23-»-4-14-4-20-24-20-30-11-17-39-4-21-20-19-36-27-5-17-5-4-23-20-6-5-16-5-4-7-20-13-9-25-27-40-4-21-20-7-10-17-5-4-21-20-4-24-20-17-21-10-4-8-17-5-13-20-18-4-(-19-5-22-20-9-40-4-16-20-19-10-29-19-20-40-4-23-20-6-22-5-17-23-37-)-4-14-4-7-9-22-25-8-4-16-4-6-5-6-16-10-4-26-11-16-17-10-40-4-23-4-21-37-24-20-8-20-4-19-20-18-10-22-5-40-4-21-20-9-27-20-9-14-24-4-14-4-19-36-27-5-10-24-4-10-15-4-21-20-9-20-17-39-4-6-5-6-16-5-4-13-5-4-24-20-17-21-25-39-4-23-20-6-5-16-5-4-13-5-4-36-6-16-25-39-4-6-5-6-16-5-4-7-4-23-24-20-22-20-19-25-4-38-4-14-4-23-20-6-5-16-5-4-13-5-4-19-10-15-39-4-25-27-7-5-24-14-17-5-4-6-5-6-16-25-4-13-5-4-36-6-16-25-4-14-4-19-10-4-21-25-31-5-10-24-39-

-22-25-27-19-25-17-5-4-6-5-6-16-5-4-19-5-4-16-20-17-10-19-14-4-21-10-22-10-9-4-5-8-10-19-24-20-18-39-

-38-4-9-5-40-38-4-8-20-7-20-22-14-24-40-38-4-21-20-21-5-17-5-23-34-39-4-19-10-4-20-24-21-14-22-5-36-23-34-39-4-14-40-38-4-8-20-7-20-22-14-24-40-38-4-21-37-24-34-4-7-10-9-11-22-4-13-5-16-7-5-23-16-14-4-38-4-35-24-20-4-24-5-16-39-4-14-4-5-21-21-5-22-5-24-4-38-4-35-24-20-4-9-10-15-23-24-7-14-24-10-17-34-19-20-4-7-10-22-19-20-39-4-7-23-11-40-38-4-8-20-7-20-22-14-24-40-38-4-19-5-27-20-9-14-24-23-37-4-7-4-7-5-19-19-20-15-4-16-20-18-19-5-24-10-39-4-7-10-9-14-24-10-4-18-10-19-37-4-7-4-18-14-17-14-28-14-36-39-

-19-25-40-4-19-5-22-20-9-40-4-16-20-19-10-29-19-20-40-4-5-27-19-25-17-39-

-38-4-5-4-30-25-6-5-2-4-38-4-23-21-22-5-30-14-7-5-36-24-39-

-38-4-21-22-20-4-30-25-6-25-40-38-4-8-20-7-20-22-14-24-40-38-4-19-14-29-10-8-20-4-19-10-4-13-19-5-36-4-14-4-7-10-9-5-24-34-4-19-10-4-7-10-9-5-36-40-4-5-4-20-23-24-5-17-34-19-20-10-4-38-4-35-24-20-4-24-5-16-39-4-7-10-9-14-24-10-4-18-10-19-37-40-4-16-5-13-19-14-24-10-39-

-19-25-40-4-25-7-10-17-14-4-6-5-6-16-25-39-

-23-19-20-7-5-4-7-13-37-17-4-5-8-10-19-24-4-23-20-6-5-29-14-31-25-4-23-7-20-36-40-4-23-19-20-7-5-4-24-16-19-25-17-4-10-11-4-19-20-23-20-18-4-7-4-23-17-10-9-33-40-4-23-16-5-13-5-17-4-«-21-23-»-4-14-4-20-24-20-30-11-17-39-

-21-20-7-10-17-5-4-23-20-6-5-29-14-31-5-4-8-17-5-13-20-18-40-4-21-20-19-36-27-5-17-5-4-21-25-23-24-20-15-4-7-20-13-9-25-27-4-14-4-7-9-22-25-8-4-16-4-8-22-5-12-9-5-19-14-19-25-4-25-21-22-5-7-9-20-18-25-4-21-20-9-27-20-9-14-24-39

ткнул этот человек собачку свою в следы возле двери, сказал «пс» и отошёл. понюхала собака воздух, повела по толпе глазом (народ, конечно, собрался) и

вдруг к бабке фёкле, с пятого номера, подходит и нюхает ей подол. бабка за толпу. собака за юбку. бабка в сторону — и собака за ней. ухватила бабку за юбку и не пущает.
рухнула бабка на колени перед агентом.
— да,— говорит,— попалась. не отпираюсь. и,— говорит,— пять ведёр закваски — это так. и аппарат — это действительно верно. всё,— говорит,— находится в ванной комнате. ведите меня в милицию.
ну, народ, конечно, ахнул.
— а шуба? — спрашивают.
— про шубу,— говорит,— ничего не знаю и ведать не ведаю, а остальное — это так. ведите меня, казните.
ну, увели бабку.
снова взял агент собачищу свою, снова ткнул её носом в следы, сказал «пс» и отошёл.
повела собачища глазом, понюхала пустой воздух и вдруг к гражданину управдому подходит.

```
alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя—.,…!?: "


def encode(text, shift):
    result = []
    for char in text.lower():
        if char in alphabet:
            index = (alphabet.index(char) + shift) % len(alphabet)
            result.append(str(index + 1))
        else:
            result.append(char)
    return "-".join(result)


def decode(encoded_text, shift):
    result = []
    for code in encoded_text.split('-'):
        if code.isdigit():
            index = (int(code) - 1 - shift) % len(alphabet)
            result.append(alphabet[index])
        else:
            result.append(code)
    return "".join(result)
```

Задание №6

0111000010100100001100001010011010101101000011100000110011111001001010011101011101001001011010001111000110111101111101001010000110100011011111100101111101011111010001101000100011011001111111101011111101001101010001001100110110111010000111101100110011010001100101111100110111110100110101011101010101000101111011011010110110111000001110101011110101101101000010011001111000111000010101110010111111101001111110100011110001101111100101111110101111110100110011011000000101101100110010101110101110100101001111110010101110110000011101010001010010011001110110100111110011011100010101101010110001000111110011111100110101100110100101110000101000111110000111100110011010001111000110001111111101001000101111111001111011110101101101011101101001111000000111011101001011000110111100110100100100110011100000010110110010101000100011001101000111001010011111100001110011101111011000011110001010010000111111101100110010101110011010010111110011011011000000110101101100001010111001011111101000000110010001011001100101011100110111101010000100110101011001111100110100101111110100110111111000000111010100010101000000100110100111101110001101111100101111110100110111111001011100110100101000000100110101011001111100110100101111111010111110100010000111000111111000011000011010101011010110110100011110001101111100101111110100110111111000010100010110000100110101011100000101110111111100001011010100111111000110111100110100101000011010011011111100101110011010010100001101011011000010100110010101000001111010111101000000001001001110101011100110000000101100011000010100111111000110111100110100101111110000111111101001011101010001000000110110110010100100001110100011011101111011101010000010000111000101000100100111010101011010011110101111011001010101100111011110101111111000111101100000011001010101100101011100101011111010011111000101110010010011001111001010011011100000010101011000111111101011110001011100100100110011110111011001010101100111011110101111111000111101100000011001010101100101001111110000011100111010111010001101010110100111110100110111100101011111111000110010101101101010101101010110100001110110000011111001000100110011110111110111101010010101111001111111000011010101011010101101000011101100110101000101100100010000101111011000000100101000111001000111101101011101000011100011110001001100111000110001010110101100101010110011101111010111111000111101100000011001010101100001111110010111111001101101100000100010111111111010111110101111111000100011110010110011010010111000101000

001111100000100001001100111000101000110110110000001001100010100100
000101111111101011111000101010110101001011001110000111011001011100
100100111001111001100000110011000011111001111110011010110011010010
111000010100011111000011110011001101111001011000110000101000010010
011101010101101001111000110101011111000000110111110011110011101010
101101000101010001111111101011111011010111111101011100000010010011
101010101100111100000111111011010101111110000001101000000110010101
100111011110101111110001111011000000110010101011000011011001111100
100011101111011000010100110100110000111110101111101011011010111010
001101111100000111001110000101001101011101000110111110101110110011
011111101110100010000111110100011100100011110010011010101011010101
101000011101100000111110010001001100111000101000110110110000001001
100010100100000010111111011001101001011111001100001101100000000100001
001001110011110011000001100110100001011101001010010110110001101111
001101001010000000100100111010011111000011110101111111111010111100 11
001111111010011011110111010100000100001101000111100011011110111110
101100111101010000110100011011111100101110110011010001000111101011
111111000000100100001100001010011001000101101110000111101000111100
010101101011111010001101000100011011001111110110011010001100101111
100110111110100110101011101010101000101111011011010110110111000001
110101011110101101101000010010011100111100011101011101001010011111
101000111100011011110111110101100111101011111100111011010011111001
101110001010011001100101011100001010111001011111110100111111001010
100001000100001110010110011010111111010011001101100000010111101011
011010111011010011110000011101110100101100111010011111111010110000
011011111000110110000110000110100000010100011111111011101101111 0001
010100001100101011100110100101111100110110110000000100

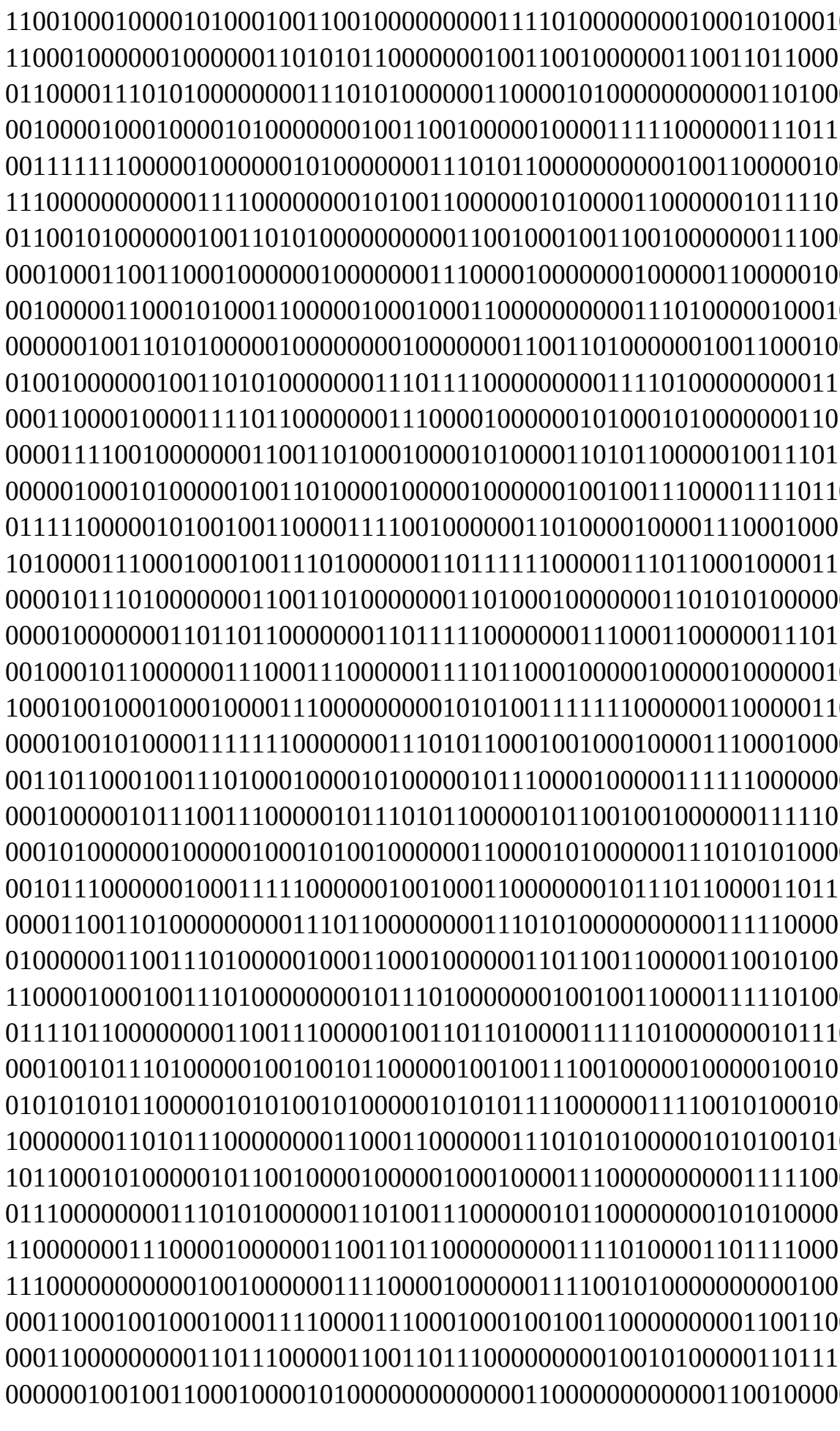
```python
from heapq import heappush, heappop
from collections import defaultdict


class HuffmanNode:
    def __init__(self, char=None, freq=0, left=None, right=None):
        self.char = char
        self.freq = freq
        self.left = left
        self.right = right

    def __lt__(self, other):
        return self.freq < other.freq

    def __str__(self):
        return str(self.char)

    def __repr__(self):
        return f"HuffmanNode({self.char}, {self.freq}, {self.left}, {self.right})"


def build_frequency_table(text):
    # Создаем таблицу частот для каждого символа в тексте
    frequency_table = defaultdict(int)
    for char in text:
        frequency_table[char] += 1
    return frequency_table


def build_huffman_tree(frequency_table):
    # Строим дерево Хаффмана на основе таблицы частот
    heap = []
    for char, freq in frequency_table.items():
        node = HuffmanNode(char, freq)
        heappush(heap, node)

    while len(heap) > 1:
        # Объединяем два узла с наименьшей частотой и создаем родительский узел
        left_node = heappop(heap)
        right_node = heappop(heap)
        parent_node = HuffmanNode(
            freq=left_node.freq + right_node.freq, left=left_node, right=right_node
        )
        heappush(heap, parent_node)
    return heap[0]


def build_encoding_table(huffman_tree):
    # Создаем таблицу кодирования на основе дерева Хаффмана
    encoding_table = {}

    def traverse(node, code):
        if node.char:
            encoding_table[node.char] = code
        else:
            traverse(node.left, code + "0")
            traverse(node.right, code + "1")

    traverse(huffman_tree, "")
    return encoding_table
```

```python
def encode_text(text, encoding_table):
    # Кодируем текст с использованием таблицы кодирования
    encoded_text = ""
    for char in text:
        encoded_text += encoding_table[char]
    return encoded_text


def decode_text(encoded_text, huffman_tree):
    # Декодируем закодированный текст с использованием дерева Хаффмана
    decoded_text = ""
    current_node = huffman_tree
    for bit in encoded_text:
        if bit == "0":
            current_node = current_node.left
        else:
            current_node = current_node.right

        if current_node.char:
            decoded_text += current_node.char
            current_node = huffman_tree

    return decoded_text


def huffman_encoding(text):
    # Основная функция для кодирования текста с использованием кода Хаффмана
    frequency_table = build_frequency_table(text)
    huffman_tree = build_huffman_tree(frequency_table)
    encoding_table = build_encoding_table(huffman_tree)
    encoded_text = encode_text(text, encoding_table)
    return encoded_text, huffman_tree


def huffman_decoding(encoded_text, huffman_tree):
    # Основная функция для декодирования закодированного текста с использованием кода Хаффмана
    decoded_text = decode_text(encoded_text, huffman_tree)
    return decoded_text
```

Задание №7

100001000101000011101010000111101100010000111000011101100100000100
010011011000100001010000111110100010000100010000010001000111100001
101011000011101110000111110100001100101000011010110000111010001000
001000100000110000111110100001100011000011000010001000111100001110
101000100001110000000000100101000011001010000111110100010011100010
000100001100101000000000100101000011101110000110101100001101001000
100101110000000000111010000111110100001101111000000001000010010000
010000110100100000000000111110001000000100001110000001011001000000000
010010100001110101000011000010000110111100001100001000000000001001
010101110000111111100010000011011101100100000100001110000001000001
0000000001000100001111101000100100010001010001100001110110010111100
010000010000011111100001111101000011110110001001110100010001011000
000001100111000011000010000000001001010000000000101001000011000010
0

0000001100001000000000100101100001101111000011010010001000011100010
0010110000000001011101000011111110000000000111010000000000110010000
00010010011000000001010100001000001000000000000111100001110111000011
111110000110101001000001000011001110000111011100000000110001100001
1111010000111100001000000010100010000111101100001100001000100000001
000011111010000110100100000000101110100001110101000000010001001000
0110101100010001111000011110110000111110100000000101110100000000001
0011100001100011000100000010000000011001110001000000110001001111001
0100110000000011100110000000000111101000011010010001000000100010000
1110000110011001000000100000000010001100000000010101100001100011000
0111010100000000101110110000100100100010100011000011101010000000010
000110000000111001000100000100001111111000100111110000000000011110
0001100111000011111000100000100000000111000010000111100100001101011
0000000111010110000000010100111000000011010011000100010110000000110
10011000011100010000000000010011000000001110101000000001000101100000
0010001111000011010110000000000010011000011010110000111001100000010
00101110000000110100110000111110100000000100000000010000001000001 0001
100001100001000000010000011100000000100100110000000001100101000000010
11001100000001010010110000111111100010000111000000001000001100001000
01100000001001011100000000100110110000001010110010001001110100000001
0000011100000001011000010000000101001111000000010101001100000000100110
1100000000011111100010000001100000000000011110000000110100010000000000
0000100010000001000000000101001000000001111010100000000111001110000000
10011001000000010101011100000001001001100001111011000000010100001100
0000010000011000010001110001000101100001100101000011000010001000001
0100001110001000000001100000000100000010000000100000101000000000000110000
010000010000000101101011000000010110111100000001111010100000001100100
0100000010001011100010000111000100100110000110000100000001001111100
1011100000101010000100000100000000101000110000000000000101000000001101
0001100000011010011100000011000111100000000100100110000000110110010
00000001000011000011110110000000001110101000000001011100100010000001
000000001000100010000010000110000100001100111000011010110000111101
1000000000000111100001111001000000011011111110000000110000101000001010
0100001100000010110010000000110000101000000010001111100000000000110111
00000000101100100010000101000000011111010100000001010001110000111111
1000000010010001000100000110001001100100000000100000110000011101100
000001011101100000000000010001000011111110000111000100000000111010110
0010011101000000100000100100000000100000110000001100010000001000000001
0000001111110010000111110100000011111110100000100000000010000001000



1100100010000101000100110010000000001111010000000010001010001010000
1100010000000100000011010101100000001001100100000011001101100010000
0110000111010100000000111010100000011000010100000000000011010000001
0010000100010000101000000010011001000001000011111000000111011111100
0011111110000010000000101000000011101011000000000010011000001001000
1110000000000011110000000010100110000001010000110000001011110110000
0110010100000010011010100000000000110010001001100100000001110000100
0001000110011000100000001000000011100001000000010000011000001001010
0010000001100010100011000001000100011000000000011101000001000100111
0000001001101010000010000000010000000110011010000001001100010000011
0100100000010011010100000001110111100000000011110100000000001111010
0001100001000011110110000000111000010000001010001010000000011011001
0000111100100000001100110100010000101000011010110000001001110111000
0000010001010000001001101000010000010000001001001110000111101100010
0111110000010100100110000111100100000011010000100001110001000100001
1010000111000100010011101000000110111111000001110110001000011100000
0000101110100000001100110100000001101000100000001101010100000001000
0000100000001101101100000001101111100000001110001100000001110111110
0010001011000000111000111000000111101100010000010000010000000100000
1000100100010001000011100000000010101001111111000000110000001100000
0000100101000011111110000000111010110001001000100001110001000000011
0011011000100111010001000010100000101110000100000111111000000001101
0001000000101110011100000101110101100000101100100100000001111011100
0001010000000100000100010100100000011000010100000011101010100000000
0010111000000100011111000000100100011000000010111011000011011110000
0000110011010000000000111011000000001110101000000000001111100001101
0100000011001110100000100011000100000011011001100000110010100100000
1100001000100111010000000010111010000000100100110000111110100000001
0111101100000000110011100000100110110100001111101000000001011101100
0001001011101000000100100101100000100100111001000001000000100101000
0101010101100000010101001010000010101011110000000111100101000100111
1000000011010111000000000110001100000011101010100000101010010100000
1011000101000000101100100001000000100010000111000000000001111100001
0111000000001110101000000110100111000000101100000000101010000100000
1100000000111000010000000110011011000000000011110100001101111000100
1110000000000001001000000011110000100000001111001010000000000100110
0001100010010001000111100001110001000100100110000000000110011000100
0001100000000001101110000011001101110000000001001010000001101111011
0000001001001100010000101000000000000001100000000000001100100000100

0110101100010100011000000100100011000001100111101000000011000101000
0001010010011000100000110000000010000110000000010001110000000011100
1010000000011000010000000011001010000000011010010000000011011010000
0000011100010000000011101010000000011110010000000011111010000000010
0000000001010100000000100001110000011011011010000000100100110000011
1000110100000111001000100000000100100110000000101111110000000110000
1100001111001000000010010110100000010011101100000000100100010000001 1
0110101000000101111011000011111010000000101000101000000000110111000
0000100111110000000101000110000000111101010000110010100000000111110
0100000001111110100000010000000100000001011110100000001110101100000
1101101000001100101011000000111010101000000110000000100010000111000
0010111101010000110000100000111111011000000011000101000000000001100
110000000101010010000110100100000101000101100000001001101000101110

```python
def encode_text(text):
    # Инициализируем словарь с начальными символами
    dictionary = {chr(i): i for i in range(2**14)}
    # Инициализируем текущую последовательность символов
    sequence = ""
    # Инициализируем пустой закодированный список
    encoded_text = []
    # Проходим по всем символам в тексте
    for char in text:
        # Обновляем текущую последовательность символов
        new_sequence = sequence + char
        # Если новая последовательность уже есть в словаре, добавляем ее к текущей последовательности
        if new_sequence in dictionary:
            sequence = new_sequence
        # Иначе добавляем код текущей последовательности в закодированный список,
        # добавляем новую последовательность в словарь и обновляем текущую последовательность
        else:
            encoded_text.append(dictionary[sequence])
            dictionary[new_sequence] = len(dictionary)
            sequence = char
    # Добавляем код последней текущей последовательности в закодированный список
    encoded_text.append(dictionary[sequence])
    return encoded_text
```

```python
def decode_text(encoded_text):
    # Инициализируем словарь с начальными символами
    dictionary = {i: chr(i) for i in range(2**14)}
    # Инициализируем предыдущий код и текущую последовательность символов
    prev_code = encoded_text[0]
    sequence = dictionary[prev_code]
    # Инициализируем пустой декодированный текст
    decoded_text = sequence
    # Проходим по кодам в закодированном списке, начиная со второго кода
    for code in encoded_text[1:]:
        # Если код есть в словаре, получаем соответствующую последовательность символов
        if code in dictionary:
            sequence = dictionary[code]
        # Иначе получаем последовательность символов, добавляя предыдущую последовательность и первый символ текущей последовательности
        else:
            sequence = dictionary[prev_code] + dictionary[prev_code][0]
        # Добавляем последовательность символов к декодированному тексту и обновляем словарь
        decoded_text += sequence
        dictionary[len(dictionary)] = dictionary[prev_code] + sequence[0]
        # Обновляем предыдущий код
        prev_code = code
    return decoded_text
```

```python
def binary_representation(encoded_text):
    # Преобразуем каждый код в закодированном списке в двоичное представление и объединяем все двоичные строки
    binary_string = ''.join(format(code, '08b') for code in encoded_text)
    return binary_string
```

Вывод: благодаря проделанной работе были сформированы навыки анализа монотонных кодов и методы программирования интервального кодирования, а также приобретеены навыки сравнения способов кодирования в системах с возможностью помехоустойчивого кодирования.

Литература

1. Тюльпинова, Н. В. Алгоритмизация и программирование : учебное пособие / Н. В. Тюльпинова. — Саратов : Вузовское образование, 2019. — 200 c. — ISBN 978-5-4487-0470-3. — Текст : электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование : [сайт]. — URL: https://profspo.ru/books/80539
2. Соснин В.В. Облачные вычисления в образовании / Соснин В.В.. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 109 c. — ISBN 978-5-4486-0512-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/79705.html
3. Шаманов А.П. Системы счисления и представление чисел в ЭВМ : учебное пособие / Шаманов А.П.. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2016. — 52 c. — ISBN 978-5-7996-1719-6. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/66204.html
4. Минитаева А.М. Кодирование информации. Системы счисления. Основы логики : учебное пособие / Минитаева А.М.. — Москва : Московский государственный технический университет имени Н.Э. Баумана, 2019. — 108 c. — ISBN 978-5-7038-5244-6. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/110640.html
5. Широков А.И. Информатика: разработка программ на языке программирования Питон: базовые языковые конструкции : учебник / Широков А.И., Пышняк М.О.. — Москва : Издательский Дом МИСиС, 2020. — 142 c. — ISBN 978-5-907226-76-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/106713.html