**Задание 1.**
Создать класс CyclicCodes. Объявить и инициализировать поля
**число информационных (исходных) разрядов** k = 4 (тип: целочисленное значение)
**длина закодированного сообщения** n = 7 (тип: целочисленное значение)
**образующий многочлен** g = [1, 0, 1, 1] (тип: массив, numpy-массив или bitarray).

*(upd 2020-10-10):* **кратность обнаруживаемой ошибки** t = 1 (тип: целочисленное значение)
*(upd 2020-10-10):* **(только для необязательных заданий 8-9) длина обнаруживаемого пакета ошибок** L (тип: целочисленное значение)

**Задание 2.**
Объявить и реализовать метод класса, осуществляющий умножение входного двоичного массива длины k на образующий многочлен g(x), соответствующий заданному вектору g. Результатом работы функции должен быть вектор, соответствующий произведению.

Обратите внимание, что:
Для a(x):

$$a=[1000], a(x)=1, c(x)=g(x), c=[1011000]$$
$$a=[0100], a(x)=x, c(x)=xg(x), c=[0101100]$$
$$a=[0010], a(x)=x^2, c(x)=x^2 g(x), c=[0010110]$$
$$a=[0001], a(x)=x^3, c(x)=x^3 g(x), c=[0001011]$$

Алгоритм умножения:
1. Преобразовать входное сообщение в многочлен по принципу:
$a=[a_0 a_1 a_2 a_3] \rightarrow a(x)=a_0+a_1 x+a_2 x^2+a_3 x^3$.
2. Умножить на образующий:
$c(x)=a(x)g(x)$.
3. Преобразовать обратно в сообщение:
$c(x)=c_0+c_1 x+c_2 x^2+c_3 x^3+c_4 x^4+c_5 x^5+c_6 x^6 \rightarrow c=[c_0 c_1 c_2 c_3 c_4 c_5 c_6]$

Псевдокод для умножения:

```
    def Encode(self, a):
        c = [0000000]
        for i = 0:7
            if (a[i])
                c[i:i+4] = c[i:i+4] + g
        return c
```

**Задание 3.**
Объявить и реализовать метод класса, осуществляющий деление с остатком входного двоичного массива длины n на образующий многочлен g(x), соответствующий заданному вектору g. Результатом работы функции должен быть вектор, соответствующий остатку от деления.

Алгоритм деления с остатком:
1. Преобразовать входное сообщение в многочлен по принципу:
$c=[c_0 c_1 c_2 c_3 c_4 c_5 c_6] \rightarrow c(x)=c_0+c_1 x+c_2 x^2+c_3 x^3+c_4 x^4+c_5 x^5+c_6 x^6$.
2. Вычислить остаток от деления на образующий:
$r(x)=c(x) \bmod g(x)$.
3. Преобразовать обратно в сообщение:
$r(x)=r_0+r_1 x+r_2 x^2 \rightarrow r=[r_0 r_1 r_2]$

Псевдокод для деления с остатком:

```
def Remainder(self, c):
    rem = c
    for i = n:n-k
        if (rem[i])
            rem[i-(n - k):i+1] = rem[i-(n - k):i+1] + g
    return rem
```

**Задание 4.**
Объявить и реализовать метод класса, осуществляющий систематическое кодирование входного двоичного массива длины k . Результатом работы функции должен быть вектор, соответствующий полученному кодовому сообщению

Алгоритм систематического кодирования:
1. Преобразовать входное сообщение в многочлен по принципу:
$a = \left[ a_0 a_1 a_2 a_3 \right] \to a\left(x\right) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ .
2. Сдвинуть вправо на n-k разрядов :
$c'\left(x\right) = x^3 a\left(x\right)$.
3. Вычислить остаток от деления на образующий:
$r\left(x\right) = c'\left(x\right) \bmod g\left(x\right)$
3. Сложить сообщение со сдвигом с остатком:
$c\left(x\right) = c'\left(x\right) + r\left(x\right)$
3. Преобразовать обратно в сообщение:
$c\left(x\right) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5 + c_6 x^6 \to c = \left[ c_0 c_1 c_2 c_3 c_4 c_5 c_6 \right]$

```
def EncodySys(self, a):
    c = [0000000]
    c[n-k : n] = a
    r = Remainder(c)
    c[0 : n-k] = r
    return c
```

*(upd. 2020-10-10)* **Задание 5 .**
Объявить и реализовать методы класса, осуществляющие формирование таблицы синдромов для всех ошибок кратности не выше **t**.

Алгоритм формирования таблицы синдромов (наивный):
1. Создать пустой словарь.
2. Для каждого кода длины **n**:
    2.1 Вычислить вес Хэмминга (wt) этого кода.
    2.2 Если вес Хэмминга не превышает **t** , добавить в словарь этот код с ключом равным остатку от деления на образующий многочлен**.**

**В результате выполнения алгоритма в словаре в качестве элементов должны находиться все векторы ошибок длины не более t и один нулевой вектор. В качестве ключей будут выступать остатки (синдромы).**

$c'\left(x\right) = x^3 a\left(x\right)$.
3. Вычислить остаток от деления на образующий:
$r\left(x\right) = c'\left(x\right) \bmod g\left(x\right)$
3. Сложить сообщение со сдвигом с остатком:
$c\left(x\right) = c'\left(x\right) + r\left(x\right)$
3. Преобразовать обратно в сообщение:

$$c(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5 + c_6 x^6 \rightarrow c = \left[ c_0 c_1 c_2 c_3 c_4 c_5 c_6 \right]$$

Псевдокод для формирования таблицы синдромов:

```
    def MakeTable(self):
        syndromes = {}
        for err = '0000000' : '1111111'
            if wt(err) <= t
                synd = Remainder(err)
                syndromes[synd] = err
        return syndromes
```

**Задание 6.**
Объявить и реализовать методы класса, осуществляющие кодирование и декодирование файлов на диске с обнаружением ошибки посредством циклических кодов с использованием таблицы синдромов из задания 5.

**Задание 7.**
Убедиться в правильности работы методов разработанного класса для циклического кода с параметрами:
**k = 7;**
**n = 15;**
**g = [1, 0, 0, 0, 1, 0, 1, 1, 1];**
**t = 2.**

**Задание 8 (необязательное).**
Модифицировать метод формирования таблицы синдромов таким образом, чтобы иметь возможность (опционально) формировать таблицу синдромов из пакетов ошибок длины не более **L** (более строгое ограничение, чем просто вес Хэмминга).

**Задание 9 (необязательное).**
Убедиться в правильности работы методов разработанного класса для циклического кода с параметрами:
**k = 9;**
**n = 15;**
**g = [1, 1, 1, 1, 0, 0, 1];**
**L = 2.**