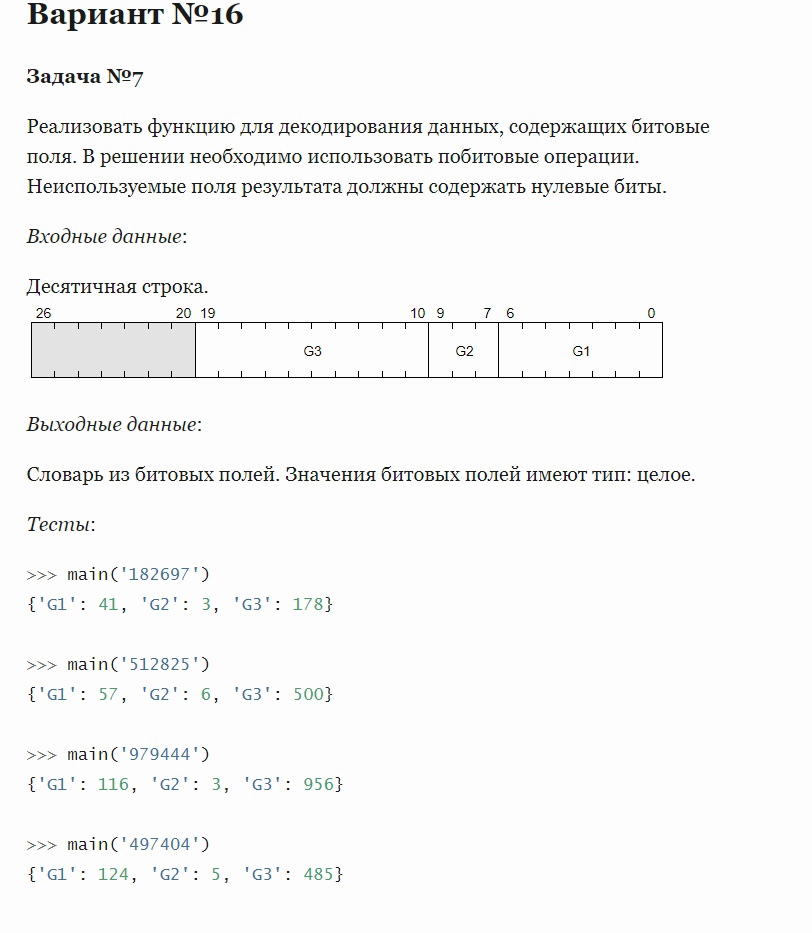


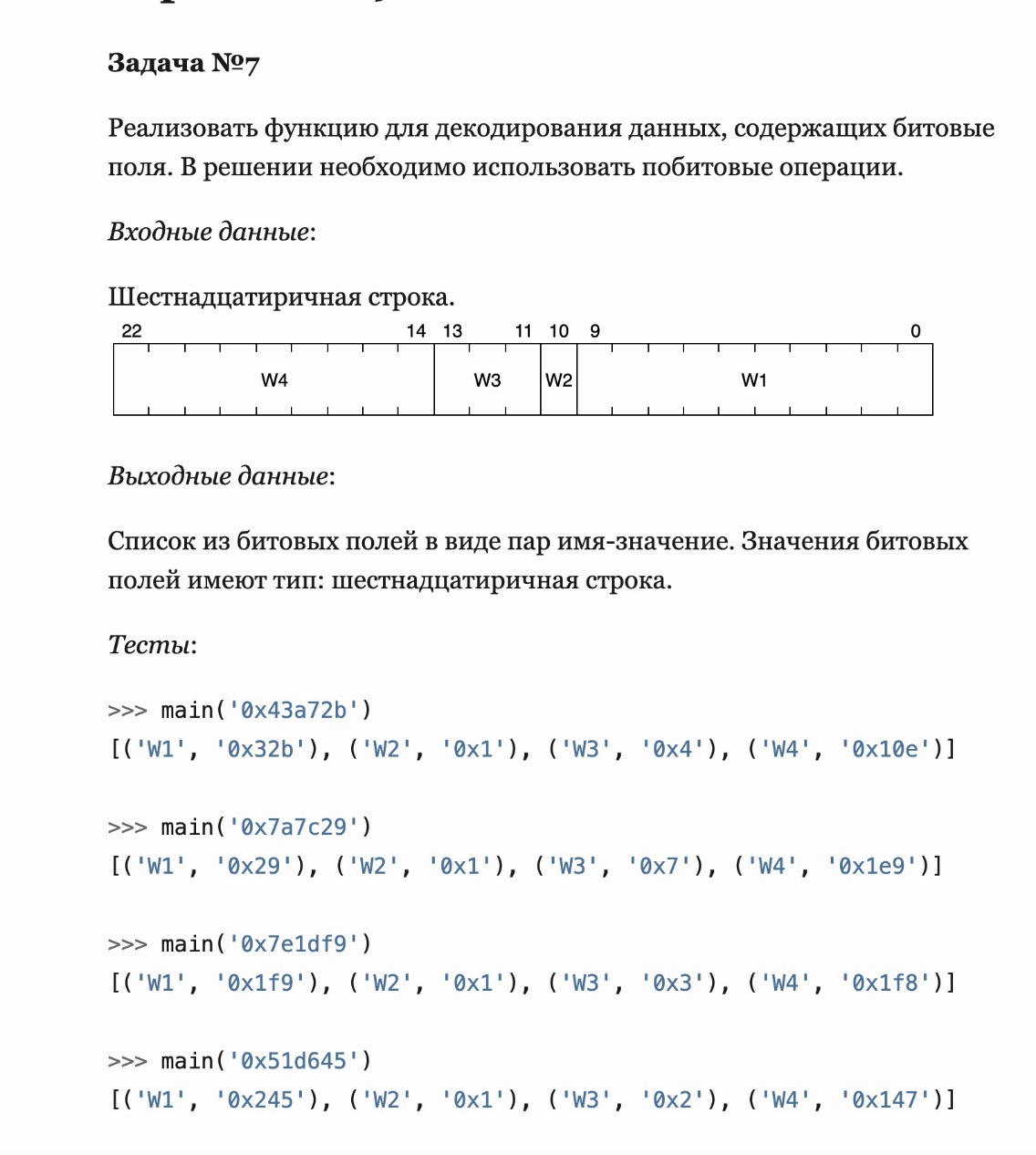
# Входные данные: словарь из битовых полей.  
# Значения битовых полей имеют тип: десятичная строка  
  
#Выходные данные: Десятичная строка  
  
def main(s):  
 k1 = int(s['o1'])  
 k3 = int(s['o3'])  
 k4 = int(s['o4'])  
 d = k1 | (k3 << 12) | (k4 << 22)  
 return str(d)  
  
print(main({'o1':'48', 'o3': '353', 'o4': '3'}))



# Входные данные: десятичная строка  
  
# Выходные данные: Словарь из битовых полей.

# Значения битовых полей имеют тип: целое

def main(x):  
 x = int(x)  
  
 g1mask = 0b1111111  
 g2mask = 0b1110000000  
 g3mask = 0b11111111110000000000  
  
 dic = {  
 'G1': g1mask & x,  
 'G2': (g2mask & x) >> 7,  
 'G3': (g3mask & x) >> 10,  
 }  
  
 return dic  
  
print(main('182697'))



# Входные данные: шестнадцатеричная строка  
  
# Выходные данные: Список из битовых полей в виде пар имя-значение

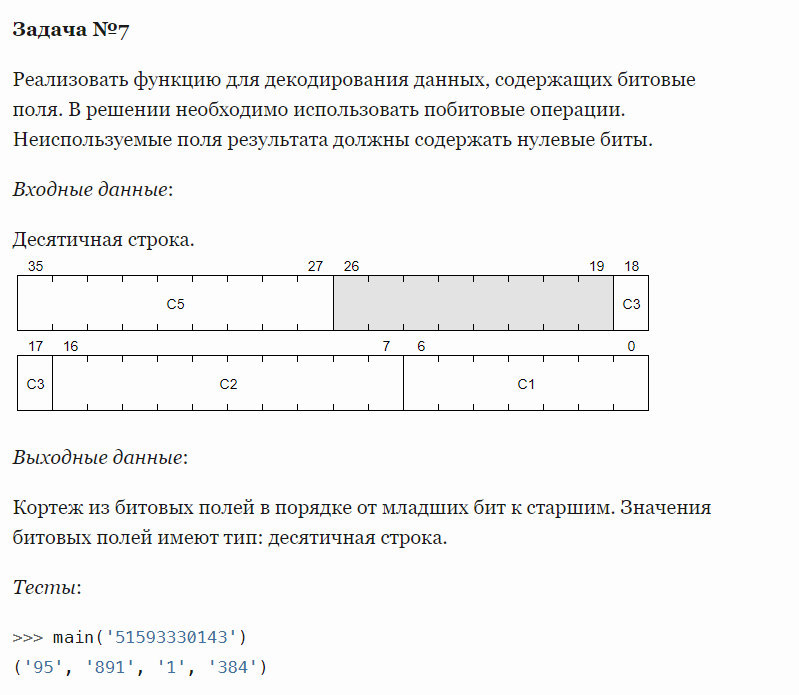
# Значения битовых полей имеют тип: шестнадцатеричная строка

def main(x):  
 e = x & (0b1 << 31)  
 d = x & (0b11111 << 26)  
 c = x & (0b1111 << 22)  
 b = x & (0b111111111 << 13)  
 a = x & 0b1111111111111  
  
 return (d >> 13) | (b >> 13) | (e >> 13) | (a << 19) | (c >> 13)  
  
print(hex(main('0x43a72b')))  
print(hex(main('0x7a7c29')))  
print(hex(main('0x7e1df9')))  
print(hex(main('0x51d645')))

# Входные данные: шестнадцатеричная строка  
  
# Выходные данные: Словарь из битовых полей в виде пар имя-значение

# Значения битовых полей имеют тип: шестнадцатеричная строка

def main(x: str) -> dict:  
 X: int = int(x, 16)  
 D: dict = {}  
 D['S1'] = hex(X & 0xf)  
 D['S2'] = hex(X >> 4 & 0b1111111)  
 D['S3'] = hex(X >> 11 & 0b1111111111)  
 D['S4'] = hex(X >> 21 & 0b1111111)  
 D['S6'] = hex(X >> 31 & 0b1)  
 return D  
  
  
print(main('0xa5fc49f'))  
# {'S1': '0xf', 'S2': '0x49', 'S3': '0x3f8', 'S4': '0x52', 'S6': '0x0'}  
# {'s1': '0xf', 's2': '0x49', 's3': '0x3f8', 's4': '0x52', 's6': '0x0'}  
print(main('0xc2bee1a5'))  
# {'S1': '0x5', 'S2': '0x1a', 'S3': '0x3dc', 'S4': '0x15', 'S6': '0x1'}  
print(main('0x4db952d6'))  
# {'S1': '0x6', 'S2': '0x2d', 'S3': '0x32a', 'S4': '0x6d', 'S6': '0x0'}  
print(main('0x2fc9ef62'))  
# {'S1': '0x2', 'S2': '0x76', 'S3': '0x13d', 'S4': '0x7e', 'S6': '0x0'}



# Входные данные: десятичная строка  
  
# Выходные данные: Кортеж из битовых полей в порядке от младших бит к старшим.

# Значения битовых полей имеют тип: десятичная строка

def main(s):  
 i = int(s)  
 c1 = 0b1111111 & i  
 c2 = 0b1111111111 & (i >> 7)  
 c3 = 0b11 & (i >> 17)  
 c5 = 0b111111111 & (i >> 27)  
 return tuple(map(str, (c1, c2, c3, c5)))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def main(\*r):  
 s = ({'PIC', 'DART', 'COBOL', 'NUMPY'},  
 {'PIC', 'DART', 'COBOL', 'HY'},  
 {'PIC', 'DART', 'CUDA'},  
 {'PIC', 'DART', 'XS', 'NUMPY'},  
 {'PIC', 'DART', 'XS', 'HY'},  
 {'PIC', 'CSON'},  
 {'OCAML', 'DART'},  
 {'OCAML', 'CSON', 'NUMPY', 'COBOL'},  
 {'OCAML', 'CSON', 'NUMPY', 'CUDA'},  
 {'OCAML', 'CSON', 'NUMPY', 'XS'},  
 {'OCAML', 'CSON', 'HY'},  
 {'EC'})  
 s1 = set(\*r)  
 return [i for i in range(len(s)) if not (len(s[i] - s1))][0]  
  
  
print(main(['CSON', 'CUDA', 'NUMPY', 1994, 'OCAML']))  
print(main(['CSON', 'COBOL', 'NUMPY', 2000, 'EC']))  
print(main(['DART', 'CUDA', 'HY', 2000, 'PIC']))  
print(main(['CSON', 'XS', 'HY', 2000, 'PIC']))  
print(main(['CSON', 'COBOL', 'NUMPY', 2000, 'OCAML']))