```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
# zy6 js encode
# 初始变换序列，可以更改
zy6_BLOCK_SIZE = int(384 / 8)
m_zy64 = [63, 0, 34, 17, 58, 23, 16, 2, 59, 55, 56, 7, 47, 61, 8, 4, 38, 25, 48, 3, 37, 41, 1, 32, 39,
57, 33, 30, 18, 45, 14,
          28, 15, 60, 5, 40, 43, 53, 10, 20, 49, 35, 62, 31, 6, 24, 22, 26, 46, 29, 36, 9, 11, 52, 44,
13, 27, 54, 19, 50, 51, 12, 42, 21]


class zy_st_CLASS:
    def __init__(self):
        self.m_tot_len = 0
        self.m_len = 0
        self.m_zy_c_64 = []
        for i in range(len(m_zy64)):
            self.m_zy_c_64.append(m_zy64[i])
        # self.m_zy_c_64 = [63, 0, 34, 17, 58, 23, 16, 2, 59, 55, 56, 7, 47, 61, 8, 4, 38, 25, 48, 3,
37, 41, 1, 32, 39, 57, 33, 30, 18, 45, 14,
         #    28, 15, 60, 5, 40, 43, 53, 10, 20, 49, 35, 62, 31, 6, 24, 22, 26, 46, 29, 36, 9, 11, 52,
44, 13, 27, 54, 19, 50, 51, 12, 42, 21]
        self.m_block = [0]*(2 * zy6_BLOCK_SIZE)
        self.m_zy_p = [0]*64


# 复制字符串
def memcpy(s, k, d, m, i):
    s[k:k+i] = d[m:m+i]

# 实现char字符到 6 bit的转换


def To64(c, d):
    # for i in range(zy6_BLOCK_SIZE):
     #   if c[i].isalpha():
      #      c[i] = ord(c[i])
    for i in range(int(zy6_BLOCK_SIZE/3)):
        m = i * 3
        d.append(c[m] >> 2)
        d.append(((c[m] & 0x3) << 4) + (c[m + 1] >> 4))
        d.append(((c[m + 1] & 0xf) << 2) + (c[m + 2] >> 6))
        d.append(c[m + 2] & 0x3f)
    return d


def update(z_st, message, len):
    tmp_len = zy6_BLOCK_SIZE - z_st.m_len
    rem_len = len if len < tmp_len else tmp_len

    memcpy(z_st.m_block, z_st.m_len, message, 0, rem_len)
    if z_st.m_len + len < zy6_BLOCK_SIZE:
        z_st.m_len += len
        return

    new_len = len - rem_len
    block_nb = int(new_len / zy6_BLOCK_SIZE)
    shifted_message = [0]*(len - rem_len)
```

```
    memcpy(shifted_message, 0, message, rem_len, len-rem_len)
    transform(z_st, z_st.m_block, 1)
    transform(z_st, shifted_message, block_nb)
    rem_len = new_len % zy6_BLOCK_SIZE

    memcpy(z_st.m_block, 0, shifted_message,
           block_nb * zy6_BLOCK_SIZE, rem_len)
    z_st.m_len = rem_len
    z_st.m_tot_len += (block_nb + 1) * zy6_BLOCK_SIZE


def final(z_st):
    block_nb = (1 + ((zy6_BLOCK_SIZE - 9) < (z_st.m_len % zy6_BLOCK_SIZE)))
    pm_len = block_nb * zy6_BLOCK_SIZE
    for i in range(z_st.m_len, zy6_BLOCK_SIZE * 2):
        z_st.m_block[i] = 0

    z_st.m_block[z_st.m_len] = 0x80
    transform(z_st, z_st.m_block, block_nb)
    pt = 0

    for i in range(64):
        z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[i] + z_st.m_zy_c_64[pt]) % 64
        pt = m_zy64[pt]
        z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[i] + z_st.m_zy_c_64[pt]) % 64
        pt = m_zy64[pt]
        z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[i] + z_st.m_zy_c_64[pt]) % 64
        pt = m_zy64[pt]
        z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[i] + z_st.m_zy_c_64[pt]) % 64
        pt = m_zy64[pt]


def transform(z_st,  message,  block_nb):
    sub_block = [0]*96
    pt = 0
    for i in range(block_nb):
        memcpy(sub_block, 0, message, i * zy6_BLOCK_SIZE, zy6_BLOCK_SIZE)
        d = []
        # 组成64卦
        To64(sub_block, d)
        for ti in range(int((zy6_BLOCK_SIZE * 4) / 3)):

            if d[ti] > 0:
                pt = ti

                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + d[ti]) % 64
                pt = m_zy64[pt]

                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + d[ti]) % 64
                pt = m_zy64[pt]
                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + d[ti]) % 64
                pt = m_zy64[pt]

                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + d[ti]) % 64

                pt = d[ti]

                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64
```

```
                pt = m_zy64[pt]
                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64
                pt = m_zy64[pt]
                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64
                pt = m_zy64[pt]
                z_st.m_zy_c_64[pt] = (z_st.m_zy_c_64[pt] + m_zy64[pt]) % 64


# 变换


def Tostring64(m_zy6_st):
    letters = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ=-"
    ss = ""
    for i in range(64):
        ss += letters[m_zy6_st.m_zy_c_64[i]]
    return ss


def Tostring64HEX(m_zy6_st):
    j = 0
    for i in range(int(64/4)):
        m_zy6_st.m_zy_p[j * 3] = (m_zy6_st.m_zy_c_64[i * 4]
                                  << 2) + (m_zy6_st.m_zy_c_64[i * 4 + 1] >> 4)
        m_zy6_st.m_zy_p[j * 3 + 1] = ((m_zy6_st.m_zy_c_64[i * 4 + 1] & 0xf)
                                      << 4) + (m_zy6_st.m_zy_c_64[i * 4 + 2] >> 2)
        m_zy6_st.m_zy_p[j * 3 + 2] = ((m_zy6_st.m_zy_c_64[i * 4 + 2] & 0x3)
                                      << 6) + m_zy6_st.m_zy_c_64[i * 4 + 3]
        j = j + 1

    letters = "0123456789abcdef"
    ss = ""
    for i in range(48):
        c = m_zy6_st.m_zy_p[i]
        ss += letters[(c >> 4) & 0xf]
        ss += letters[c & 0xf]
    return ss


def Tostring64BG(m_zy6_st):
    letters ="䷀䷁䷂䷃䷄䷅䷆䷇䷈䷉䷊䷋䷌䷍䷎䷏䷐䷑䷒䷓䷔䷕䷖䷗䷘䷙䷚䷛䷜䷝䷞䷟䷠䷡䷢䷣䷤䷥䷦䷧䷨䷩䷪䷫䷬䷭䷮䷯䷰䷱䷲䷳䷴䷵䷶䷷䷸䷹䷺䷻䷼䷽䷾䷿"
    ss = ""
    for i in range(64):
        ss += letters[m_zy6_st.m_zy_p_64[i]]
    return ss


# 将字符串格式化为UTF8编码的字节
def writeUTF(str, isGetBytes):
    back = []
    byteSize = 0
    for i in range(len(str)):
        code = ord(str[i])
        if 0x00 <= code and code <= 0x7f:
            byteSize += 1
            back.append(code)
        elif 0x80 <= code and code <= 0x7ff:
            byteSize += 2
```

```
            back.append((192 | (31 & (code >> 6))))
            back.append((128 | (63 & code)))
        elif (0x800 <= code and code <= 0xd7ff) or (0xe000 <= code and code <= 0xffff):
            byteSize += 3
            back.append((224 | (15 & (code >> 12))))
            back.append((128 | (63 & (code >> 6))))
            back.append((128 | (63 & code)))

    for i in range(len(back)):
        back[i] &= 0xff

    if isGetBytes:
        return back

    if byteSize <= 0xff:
        return [0, byteSize].extend(back)
    else:
        return [byteSize >> 8, byteSize & 0xff].extend(back)


def updatestr(m_zy6_st, str):
    qstr = writeUTF(str, 1)
    return update(m_zy6_st, qstr, len(qstr))


def Zy6String(ss):
    zy_st = zy_st_CLASS()
    updatestr(zy_st, ss)
    final(zy_st)
    return Tostring64(zy_st)


def Zy6StringHEX(ss):
    zy_st1 = zy_st_CLASS()
    updatestr(zy_st1, ss)
    final(zy_st1)
    return Tostring64HEX(zy_st1)


test_ss = input()
print(Zy6String(test_ss))
print(Zy6StringHEX(test_ss))
#test_ss = '举例分析？.'
#print(Zy6String(test_ss))
#print(Zy6StringHEX(test_ss))
```