

实现一种整数编码方法，使得待编码的数字越小，编码后所占用的字节数越小。

编码规则如下：

- 1、编码时 7 位一组，每个字节的低 7 位用于存储待编码数字的补码。
- 2、字节的最高位表示后续是否还有字节，置 1 表示后面还有更多的字节，置 0 表示当前字节为最后一个字节。
- 3、采用小端序编码，低位和低字节放在低地址上。
- 3、编码结果按 16 进制数的字符格式输出，小写字母需转换为大写字母。

输入描述：

输入的为一个字符串表示的非负整数

输出描述：

输出一个字符串，表示整数编码的 16 进制码流

补充说明：

待编码的数字取值范围为 $[0, 1 \ll 64 - 1]$

示例 1

输入：

0

输出：

00

说明：

输出的 16 进制字符，不足两位的前面补 0，如 00、01、02。

示例 2

输入：

100

输出：

64

说明：

100 的二进制表示为 0110 0100，只需要一个字节进行编码；

字节的最高位置 0，剩余 7 位存储数字 100 的低 7 位（110 0100），所以编码后的输出为 64。

示例 3

输入：

1000

输出：

E807

说明：

1000 的二进制表示为 0011 1110 1000，至少需要两个字节进行编码；

第一个字节最高位置 1，剩余的 7 位存储数字 1000 的第一个低 7 位（110 1000），所以第一个字节的二进制为 1110 1000，即 E8；

第二个字节最高位置 0，剩余的 7 位存储数字 1000 的第二个低 7 位（000 0111），所以第一个字节的二进制为 0000 0111，即 07；

采用小端序编码，所以低字节 E8 输出在前，高字节 07 输出在后。

```
import math
```

```
in10=int(input())
```

```

in2=bin(in10)
in2str=str(in2)[2::]
in2str1=in2str[::-1]
n=math.ceil(len(in2str)/7)
ans=""
for i in range(n):
    if i==n-1:
        # start=-1*i
        # end=0
        temp=in2str1[7*i::][::-1]
    else:
        start=7*i
        end=7*(i+1)
        temp=in2str1[start:end][::-1]
    if i==n-1:
        while len(temp)<8:
            temp='0'+temp
        t1 = temp[:4]
        t2 = temp[4:]
        tt1 = int(t1, base=2)
        tt1 = hex(tt1)[2:].upper()
        ans = ans + tt1
        tt2 = int(t2, base=2)
        tt2 = hex(tt2)[2:].upper()
        ans=ans+tt2
    else:
        temp = '1' + temp
        t1=temp[:4]
        t2=temp[4:]
        tt1=int(t1,base=2)
        tt1 = hex(tt1)[2:].upper()
        ans=ans+tt1
        tt2 = int(t2, base=2)
        tt2 = hex(tt2)[2:].upper()
        ans=ans+tt2

print(ans)

```