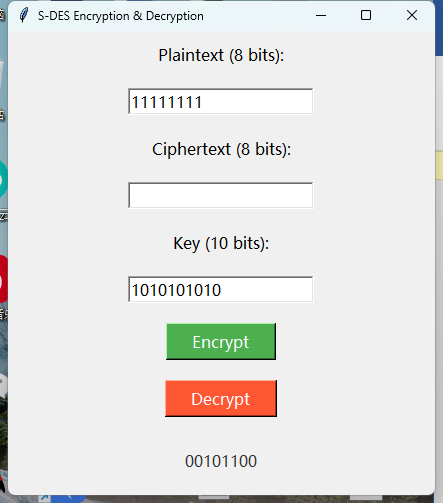
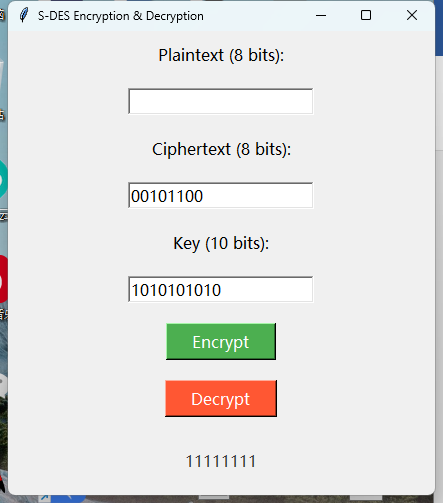
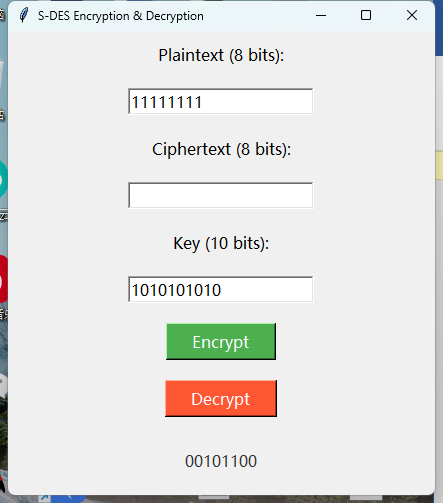
关卡1（使用1.py）：

我们使用明文：11111111，密钥：1010101010  
得到密文：00101100，基本测试完成。  


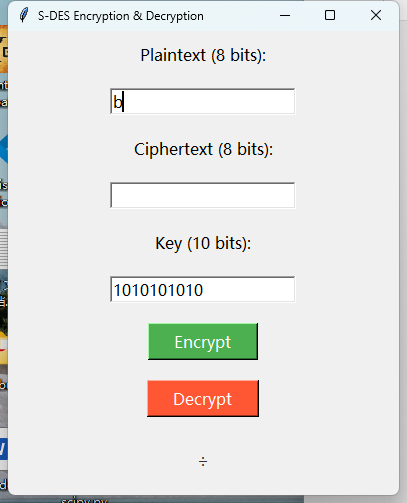
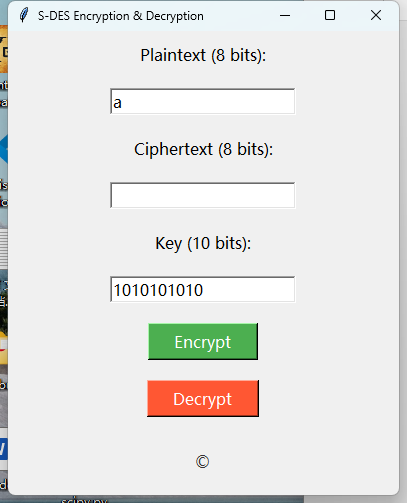
关卡2（使用1.py）：  
验证明文，密文是否能够正确互相对应：  


可以看到明文11111111，密钥1010101010，可以得到密文为：00101100；

而密文00101100，密钥1010101010，可以得到明文11111111。交叉测试完成。

**注：**我们同时还打印了算法运行的过程，可以供检查使用，每一步计算是否正确。

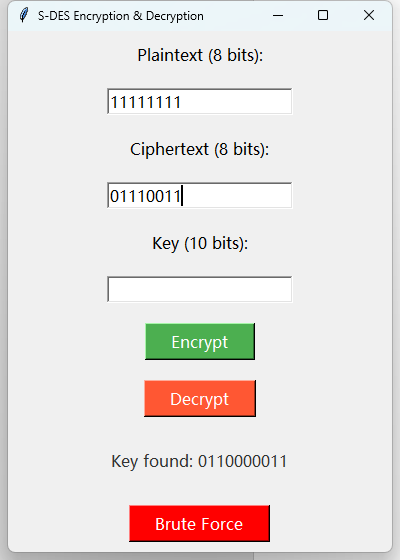
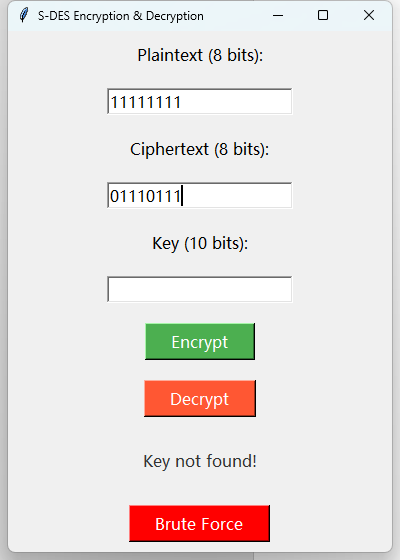
关卡3（使用3.py）：



我们使用了a作为明文，使用1010101010作为密钥，可以得到©

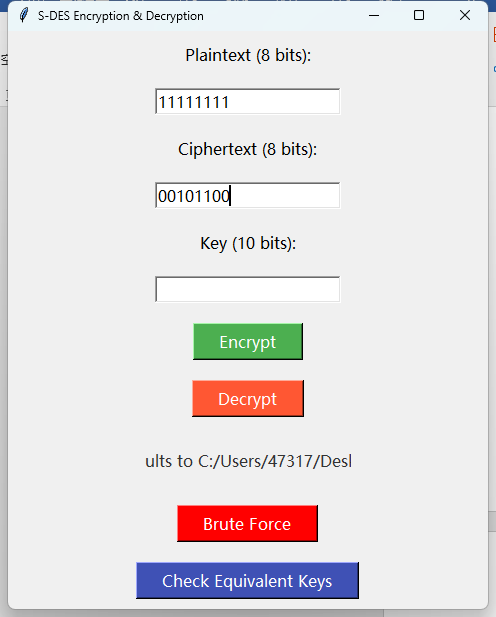
我们使用了b作为密文，使用1010101010作为密钥，得到了➗。

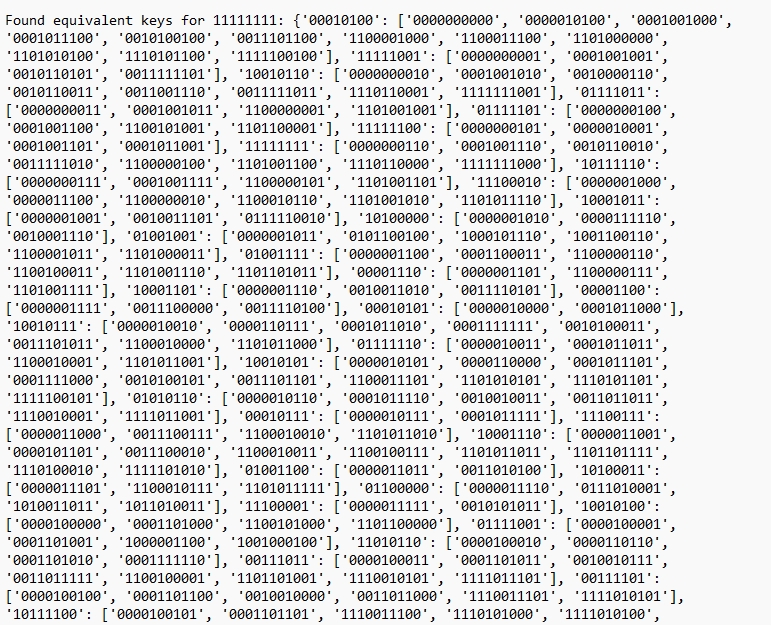
**注：**我们仅能输入一个ascii码。

关卡4（4.py）：  
 

我们使用多线程，暴力破解法，将每一个密钥带入，判断是否能正常加密与解密。若能正常加密与解密，则输出第一个，若不能则反应无法找到密钥。

**注：**可能会有多个密钥的出现，在本关卡，我们仅考虑第一个找到的密钥。

关卡5（5.py）：  
 

输入一组明文与密钥以后，我们可以得到一系列的密钥，然后我们将密钥储存在一个txt中：  
得到如下结果：  


注：不止存在一个密钥，且仅选取一部分展示。

同时为了回答问题2，我们设计了如下实验：

1.遍历每一个可能的明文分组。

2.对于每一个明文分组，加密它使用每一个可能的密钥。

3.对于每一个密文，检查是否有其他密钥也产生了相同的密文。

得到如下结果：  
Collision found!

For plaintext 00000000, keys 0000010111 and 0000000011

produce the same ciphertext 00000101.

**注：**这种方法在时间上可能是昂贵的，因为它需要执行大约256×1024次S-DES加密操作。