In this task, you are given a plaintext and a ciphertext, and your job is to find the key that is used for the encryption. You do know the following facts:

- The `aes-128-cbc` cipher is used for the encryption.
- The key used to encrypt this plaintext is an English word shorter than 16 characters; the word can be found from a typical English dictionary. Since the word has less than 16 characters (i.e. 128 bits), pound signs (`#`: hexadecimal value is `0x23`) are appended to the end of the word to form a key of 128 bits.

Your goal is to write a program to find out the encryption key. You can download a English word list from the Internet. We have also linked one on the web page of this lab. The plaintext, ciphertext, and IV are listed in the following:

```
Plaintext (total 21 characters): This is a top secret.
Ciphertext (in hex format): 764aa26b55a4da654df6b19e4bce00f4
                            ed05e09346fb0e762583cb7da2ac93a2
IV (in hex format):         aabbccddeeff00998877665544332211
```

You need to pay attention to the following issues:

- If you choose to store the plaintext message in a file, and feed the file to your program, you need to check whether the file length is 21. If you type the message in a text editor, you need to be aware that some editors may add a special character to the end of the file. The easiest way to store the message in a file is to use the following command (the `-n` flag tells `echo` not to add a trailing newline):

  ```
  $ echo -n "This is a top secret." > file
  ```

- In this task, you are supposed to write your own program to invoke the crypto library. No credit will be given if you simply use the `openssl` commands to do this task. Sample code can be found from the following URL:

  ```
  https://www.openssl.org/docs/man1.0.2/crypto/EVP_EncryptInit.html
  ```

- When you compile your code using `gcc`, do not forget to include the `-lcrypto` flag, because your code needs the `crypto` library. See the following example:

  ```
  $ gcc -o myenc myenc.c -lcrypto
  ```