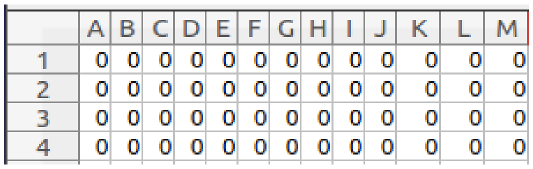
YSC2221 Exercise 04

Question 1: Treasure Hunt – X marks the spot! (30 Marks)

You're on a quest for treasure, and you've been given a 300 row x 300 column treasure map in the form of a CSV file. An example of such file is in `treasure.csv`.
The file looks something like this:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0

Within the treasure map file, there's a particular cell which marks the place where the treasure is buried. The cell is marked with the string '**x**'.

Your task is to write a function **`find_treasure`** that:

- takes a string `file_path_map` (e.g. `'treasure.csv'`) and reads the CSV file into a **2D List**. (Hint: use the csv library!)
- returns a **tuple** `(row, column)` of the coordinates of the treasure, e.g. `(111, 222)` **.

<u>No marks will be given if you just output the coordinate of the treasure via other means.</u>**

Question 2: Decrypting the Map – Practice! (30 Marks)

Congratulations! You've found the spot and have dug up the spot only to find... another piece of paper! Rats! It seems to be encrypted in some form of gibberish language (in the file 'message.txt')...

```
"qa mqtbppd vjqtu mghto! esbtr dgh mgz mqtoqtu aj gs rqto
xezbtujz. q ahxe ejpp dgh, qex whqej vgzqtu sbfqtu bpp esj
ljbpes qt esj lgzpo. qm dghzj pggrqtu mgz qe, q vhzqjo b aby
eg qe bzghto lsjzj dghzj xebtoqtu tgl! zjajavjz esghus:
ljbpes qxte jfjzdesqtu qt esj lgzpo!"
```

But wait! Not all hope is lost! You spot a **letter substitution translation guide** at the bottom right corner of the treasure map, which is in the form of a CSV file. An example of such file is in `translation.csv`.

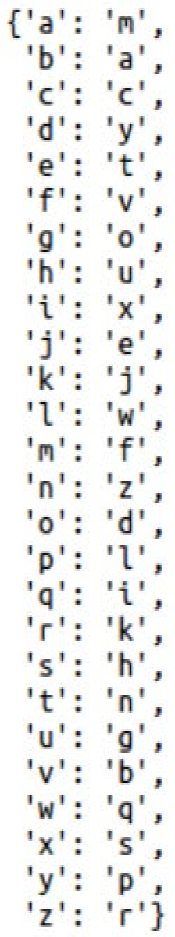
The translation guide consists of two columns: the first column `cryptic_language` contains letters which corresponds to that in the second column `English`.

Your task is to write a function **`read_translation_guide_into_dictionary`** that:

- takes a string `file_path_translation` (e.g. `'translation.csv'`) and reads the CSV file.
- returns a **dictionary** with the *key* being `cryptic_language`, and the *value* being `english`.
- An example of the returned dictionary is on the right.

Question 3: Decrypt the Map! (30 Marks)

You have your trusty translation dictionary (pun intended?) at hand, now it's time to decrypt the message! An example of such file is in `message.txt`. (You have to write your own helper function to load it into memory like Question 1, **but as a string**.)



```
{'a': 'm',
 'b': 'a',
 'c': 'c',
 'd': 'y',
 'e': 't',
 'f': 'v',
 'g': 'o',
 'h': 'u',
 'i': 'x',
 'j': 'e',
 'k': 'j',
 'l': 'w',
 'm': 'f',
 'n': 'z',
 'o': 'd',
 'p': 'l',
 'q': 'i',
 'r': 'k',
 's': 'h',
 't': 'n',
 'u': 'g',
 'v': 'b',
 'w': 'q',
 'x': 's',
 'y': 'p',
 'z': 'r'}
```

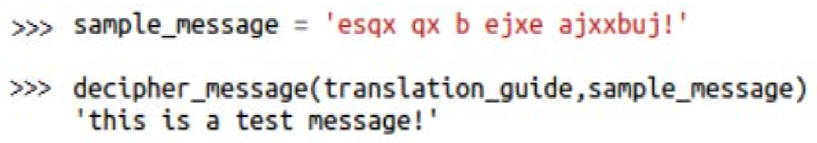
Use your skills to write a function **`decipher_message`** that takes in the following:

- first argument: a translation **dictionary** (obtained from Part A Question 2), and
- second argument: a **string** (not a file path) containing the encrypted message.

Then decrypts the encrypted message, and returns the decrypted message as a **string**.

Characters in the encrypted message which are not found in the translation dictionary (keys) can be taken as-is without having to do any substitution.

Sample Run:



```
>>> sample_message = 'esqx qx b ejxe ajxxbuj!'

>>> decipher_message(translation_guide,sample_message)
    'this is a test message!'
```

(Hint: How can you use what you've learned so far?)

Question 4: Find the Treasure...Map! (40 Marks)

You search the nearby area and manage to find another map! This is it! You're close to being rich beyond belief! But wait, it's encrypted again! An example of such file is in `encrypted_map.txt`.

Your task is to write a function **`decrypt_map`** that:

- takes a string containing the file path to an encrypted map (e.g. '`encrypted_map.txt`').
- takes a string containing the file path to a translation guide (e.g. '`map_code.csv`').
- decrypts this encrypted map using the translation guide.
- writes this decrypted map to a file called **`decrypted_map.txt`**.

An example of `decrypted_map.txt` (using the provided `encrypted_map.txt`) is on the right.
Note that there must not be any empty newline characters at the end of `decrypted_map.txt`.

```
WWWWWWWWWW,T,WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW,WWWWWWWWWW,,,,,WWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWW,,,,,,,WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW,,TT,,WWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW,,,,,WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWW,,,WWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
```

Congratulations! You're rich!