Assignment on File + Dictionaries

Task 1:

You're a detective in the pre-futuristic 1980s. You have a robot sidekick who goes to crime scenes to collect statements of witnesses in a text file. Your sidekick collects his notes like this:

statements.txt

```
Suspect Name 1
Alibi
Suspicious behaviour by as seen witnesses
-
Suspect Name 2
Alibi 2
Suspicious behaviour by as seen witnesses 2
-
…
```

It collects the statements for each suspect in 3 lines, ends it using a - symbol and then moves on to the next suspect.

After it collects the statements, your job is to use a powerful computer algorithm to find the culprit using the descriptions and print their name and alibi to a file like this:

culprit.txt

```
<name> is the culprit because <behavior>
```

You're investigating a murder of the curator at a museum. When the guards at the museum checked the cameras, they said the culprit looked familiar to a person who came yesterday and was seen **arguing with the museum curator**. Your robot sidekick has done his job of collecting the statements from witnesses which are stored in this file. Your task is to write the computer algorithm to find the culprit and print their name in a **culprit.txt** file.

Continues on the next page

No input from this task.

Sample Input	**Sample Output**
Read from file	Alice is the culprit because Was seen arguing with the museum curator the day before the robbery

HINTS:

1. Read the lines and store each of the names, alibi, and behaviour in a dictionary. You can use a structure like this:
   ```
   {
       name1: {alibi: <value>, behavior: <value>},
       name2: {alibi: <value>, behavior: <value>},
       …….
   }
   ```
2. Check if the line has a - symbol. If yes, you can simply skip to the next statements.
3. The guards said the culprit was seen **arguing with the museum curator.** Can we do a string check on the behaviours of each k-v pair to see where we can find a similar statement from our dictionary?

TASK 2 STARTS FROM NEXT PAGE

Task 2

You are trying to create a terminal command line that will get you all/selective data from a file. However, you want to keep it very simple. You only decide to create commands that gets all the data from a file, but you also want to make sure that it can get data related to a field, and link them with other fields.

So you decided that the syntax of your command will be:

getall <column> from <text_file_name>

And it can also have the syntax of

getall <column> link <another_column> from <text_file_name>

The first command will get you all the data from a certain text file. This can be in the format of:
John Doe, 1, jdoe@gmail.com
Jane Doe, 2, jane@gmail.com
…..
This is the output for the command: **getall * from users**. If Python sees the "getall * from users", it will read all the data of the line. But if it reads "getall name from users", then it will read all the lines of the file but will only give you the names:
John Doe,
Jane Doe,
Jack Doe
….

This command is quite simple, **as it just reads line by line from the file**.

The tricky part however is the second command. The second syntax will get you only the specific data from lines as requested in the command but will give **you an output of a list of dictionaries**. If you were to have some data like:
Column1, column2, column3

And your command was: getall column1 link column2 from …,
Then you could link them like this:
{ column1: column2 }
Where column1 would be the key and column2 would be the value.

P.T.O.

Similarly, for the data:
John Doe, 1, jdoe@gmail.com,
Jane Doe, 2, jane@gmail.com

And if the command were to be: **getall name link email from users**, then the output would be:

```
[
  { “John Doe”: “jdoe@gmail.com”},
  { “Jane Doe”: “jane@gmail.com”},
]
```

Your task is to implement a Python code for **getall** that reads the input query as a string and outputs the fields according to your query. Find the users.txt file right here.

Sample Input	**Sample Output**
getall * from users	John Doe, 1, jdoe@gmail.com Jane Doe, 2, jane@gmail.com …
getall name from users	John Doe Jane Doe …
getall email from users	jdoe@gmail.com jane@gmail.com …
getall email link id from users	[{ “jdoe@gmail.com”: “1”}, { “jane@gmail.com”: “2”}, ….]
getall name link email from users	[{ “John Doe”: “jdoe@gmail.com”}, { “Jane Doe”: “jane@gmail.com”}, ….]

HINTS:
1. **Split** the command string early on.
2. The name, id, and email won't be specified in your file. Use **fixed indices to represent them (think of the name and salary problem)**.
   Ex: index 0 of the list of split strings can always be the type of query (getall/insert)

3. To check if you want to do a simple getall (Syntax 1) or a link getall (Syntax 2), why not **check if a certain string is present** in the list of split strings? (if link in ….)
4. **Don’t forget to check if there is a * or a certain column** before you do the rest of the code for Syntax 1.
5. For linking the second command, would it be possible to **<u>simply find the fixed positions and put them in a separate dictionary for link commands</u>**?