CSCA08 Exercise 7

Due: November 11, 2016. 5:00pm

This time we're working with dictionaries. In fact, we're working with a specific dictionary in all three functions: This dictionary maps a username to a list of last name, first name, e-mail, age and gender. We will build the dictionary from a file, update the dictionary, and then select elements from the dictionary that meet a certain criteria.

Building the Dictionary

Write a function named `create_dict`, that takes one parameter, an open file handle (remember ex5, this is an open file handle, not the name of a file), and returns a dictionary that maps a string to a list of strings and ints. In particular, the type contract of this function will be:
`(io.TextIOWrapper) -> dict of {str: [str, str, str, int, str]}`
This function will read a file formatted like the provided `ex7_data.txt`. That is, each line will have a username, first name, last name, age, gender (either M or F) and an e-mail address, all separated by spaces. The function will insert each person's information into a dictionary with their username as the key, and the value being a list of `[last name, first name, e-mail, age, gender]`. When the entire file has been processed, it will then return the dictionary.

Changing the Dictionary

Write a function called `update_field`, that takes 4 parameters: A dictionary in the format created by the previous function, a username, the name of a field[1] (One of: `'LAST'`, `'FIRST'`, `'E-MAIL'`, `'AGE'` or `'GENDER'`[2]), and a new value to replace the current value of the specified field. The function should not return anything, but instead mutate the dictionary as appropriate.

An example call of the function is below:

```
>>> my_dict = {'sclause':['Clause', 'Santa', 'santa@christmas.np', 450, 'M']}
>>> update_field(my_dict, 'sclause', 'AGE', 999)
>>> my_dict == {'sclause': ['Clause', 'Santa', 'santa@christmas.np', 999, 'M']}
True
```

BONUS: Selecting Elements

Write a function called `select` that takes 4 parameters, a dictionary formatted as in the previous questions, the name of a field to select, the name of a field to check [3], and a value to check for.

The function should return a `set` of all the data elements from the selected fields of people whose checked fields were equal to the checked value. That was a bit of a mouthful, so let's see an example to make it clearer. We want to select the e-mail of anyone whose gender is equal to M:

```
>>> my_dict = {'sclause':['Clause', 'Santa', 'santa@christmas.np', 450, 'M'],
        'ebunny':['Bunny', 'Easter', 'bunny@easter.hop', 99, 'M'],
        'tfairy':['Fairy', 'Tooth', 'fairy@money4teeth.net', 0, 'F']}
>>> select(my_dict, 'E-MAIL', 'GENDER', 'M')
{'santa@christmas.np', 'bunny@easter.hop'}
```

[1] You may want to write a helper function to get the index given the field name, particularly if you're going to try the bonus question

[2] Make sure you tell the user that these are the options you're expecting. As long as you inform them, your code doesn't have to work with other inputs, though it still shouldn't crash.

[3] field names and rules are the same as in the previous function