IP Assignment 2

MAX MARKS: 100

Important Points:

- In this assignment, there are two modules which need to be implemented. One module is a2.py and the other module is simulation.py.

- In a2.py, there are functions which are already declared, along with their documentation. Each function documentation explains what exactly needs to be done, what parameters the function takes and what exactly the function returns. You should not change the function declaration, be it the function name or the function parameters, in any way, otherwise you would be given 0 marks straightaway.

- You cannot make any global variables in a2.py module. You should use the parameters for your logic of the function.

- You cannot import python libraries other than those which are already imported by us.

- There is a third file, data.json. You should not edit this file in any way, otherwise your program may not work properly.

While submitting the assignment on the classroom:

- For submission, you have to submit a ZIP file with 2 python files: a2.py and simulation.py. File formats such as .ipynb (of jupyter notebook) will not be considered. You must write your code in a .py file only.

- The name of the ZIP file should be a2_rollno.zip. Here, rollno should be like 2020xyz, where xyz is the last 3-digits of your roll number.

- Do not change the names of file a2.py and simulation.py.

- Functions in a2.py must not print anything. They should only return the appropriate values. The printing part needs to be handled in the simulation.py file.

- You cannot import any other python libraries. You can only import the a2.py module in the simulation.py module.

- Please read the PDF file very carefully.

TASK 1

This assignment would test your ability to use dictionaries. In the assignment folder, there are 3 files in total: a2.py, simulation.py and data.json as mentioned above.

In the data.json file, there is a List of dictionaries, where each dictionary has a format as shown below. Each dictionary represents a person's record.

Some points to note:

1. No two users can have the same id.

2. The friend_ids key denotes the IDs of the friends of the user. Note that if a user with id i is a friend of j, then in the friend_ids list of user i, j will be present and in the friend_ids list of user j, i will be present. Also, a user cannot be a friend of himself / herself.

3. The data types of each of the fields is as visible in the data.json and the example shown below.

4. If the ongoing field of education is False, only then will the percentage field exist. If it is true, it means that the person is currently studying in that institute and the final percentage is not available. Thus, the field will not exist. However, a person can be studying at multiple institutes at the same time.

5. All strings are treated as case-insensitive during searching and when any string comparisons are being performed. But, when a string needs to be printed to the screen, it should be printed in the same case with which it is stored in the records. Also, when a user gives a string which needs to be added to the records, the string should be added in the same case with which the user entered it.

```json
{
    "id": 0,
    "first_name": "Reeshu",
    "last_name": "Bhatia",
    "age": 23,
    "gender": "male",
    "address": {
        "house_no": 24,
        "block": "ABC",
        "town": "Gharoli",
        "city": "Delhi",
        "state": "Delhi",
        "pincode": 125987
    },
    "education": [
        {
            "institute": "ABCD School",
            "ongoing": false,
            "percentage": 95.12312
        },
        {
            "institute": "EFGH Institute",
            "ongoing": true
        },
        {
            "institute": "IJKL College",
            "ongoing": true
        }
    ],
    "blood_group": "O",
    "contacts": [
        "789654211",
        "286541324"
    ],
    "friend_ids": [2, 6]
}
```

There is a function **`read_data_from_file(file_path)`** in a2.py module. This function reads the data.json file and loads all of the data into a variable named records. You have to call this method to get the person records from the data.json file.

After reading the data, you have to implement all the other functions as mentioned. The functions, shown below, are used for filtering/updating the data based on specific conditions:

1. `read_data_from_file` *(The code for this function is already provided)*
2. `filter_by_first_name`
3. `filter_by_last_name`
4. `filter_by_full_name`
5. `filter_by_age_range`
6. `count_by_gender`
7. `filter_by_address`
8. `find_alumni`
9. `find_topper_of_each_institute`
10. `find_blood_donors`
11. `get_common_friends`
12. `is_related`
13. `delete_by_id`
14. `add_friend`
15. `remove_friend`
16. `add_education`

TASK 2

After implementing all the above functions, you need to simulate an interactive menu based scenario in the simulation.py module, where you would import the a2.py module and use all the above given methods.

The general flow of the program looks something like this:

- Greet the user and show them the given queries in a menu based fashion.

- Then, ask them which query they would like to perform. Take a query code as input from the user. The code can range from 1 to 16, which denote the functions list as shown above.

- If any query updates the records, then the subsequent queries must work with the updated records.

- Corresponding to the given query code and the functionality, ask the inputs from the user which would be passed as parameters to the respective function call.

  For example, if the user inputs query code as 16 which corresponds to the Add education functionality, you need to ask the below inputs from the user:

  i. `id` (INTEGER): The person id

  ii. `institute` (STRING): The institute name

  iii. `ongoing` (BOOLEAN): The ongoing value representing if the person is currently enrolled for that education

  iv. `percentage` (FLOAT): The person's score in percentage, if the person is not currently enrolled and has been allotted the score for the same.

- Ask the query codes from the user, until the user inputs the query code as -1.