Backend Developer Assignment

1. **Query String validation and Query JSON conversion**:
   a. Write a python script, which will take a string as input.
   b. The script should validate the string's syntax based on the below assumptions
   c. If the syntax is correct, the script should convert the string to json and print the same based on the below assumptions
   d. We'll also validate the readability of the code. So, kindly make the design as understandable as possible.

   String syntax:

   Query String Example: ((A=2 && B=3) || (C=4 && D=5))

   The script should be able to validate that the brackets are used in the right fashion.

   For example, (A=2 && B=3 is not a valid syntax because the matching ')' is not there.

   Conversion Example:

   Correct Input:

   Input: ((A=2 && B=3) || (C=4 && D=5))

   Output:

```
{
  "query": {
    "or": [
      {
        "and": {
          "A": 2,
          "B": 3
        }
      },
      {
        "and": {
          "C": 4,
          "D": 5
        }
      }
    ]
```

```
                    }
                  }
```

Incorrect Input:

Input: ((A=2 && B=3 || (C=4 && D=5))

Output: Syntax invalid

Assumptions:

1. You can assume that the operator used is always "=" like in "A=2"
2. You can assume that there will only be 2 terms in both inner and outer expressions.

   For example, (A=2 && B=3) consists of two terms A=2 and B=3. Likewise, the outer expression also consists of only two terms, one is (A=2 && B=3) and other is (C=4 && D=5)

3. Just as a hint, outer brackets are not mandatory. (A=2 && B=3) || (C=4 && D=5) expression is also valid and would produce the same output as shown for expression ((A=2 && B=3) || (C=4 && D=5))

2. **Django Application Development:**

a. Let us assume we are creating a Django REST based application to display Employee-Manager relationship.

b. When the REST call is made, the application would get all employees with their manager name and return in JSON format.

Example:

Employee is A -> Manager is B

Employee is B -> Manager is C

….. this continues till CEO who does not have a manager

Points to be considered:

1. Everybody is an employee in the organization including managers.
2. Every employee has a manager except the CEO who does not have a manager.
3. There are only three fields that are of interest here: Employee ID, Employee name, Manager name.

Expectations:

1. Create a Django application with one GET REST API which when called would return list of Employee ID, Employee Name, Manager Name.
   Example:

```
          [
 {
   "emp_id": 1,
   "emp_name": "A",
    "manager_name": "B"
  },
  {
    "emp_id": 2,
    "emp_name": "C",
    "manager_name": "B"
  },
………………
]
```

2. The Employee-Manager relationship should be modelled in the models file with proper primary and foreign key relationships
3. From the above point, you could write the same query using SQL syntax and place the query in a separate .sql file. This will not be executed as part of the Django application and will be used only by us to validate.
4. The data should be queried and return to the views file
5. The whole project should be sent to us so that we can execute and validate
6. SQlite should be used so that the data can be sent to us along with the project.
7. We'll also validate the readability of the code. So, kindly make the design as understandable as possible.

3. Write a python program to print a dictionary key value ordered by length of value. If name of values is same for 2 keys sort them by key.
   Primary sort by length of value secondary sort by key.

Input :

```
data = {'hello': ['doc1'], 'my': ['doc1'], 'name': ['doc1'], 'is': ['doc1', 'doc2'], 'james': ['doc1', 'doc2'],
    'a': ['doc2'], 'developer': ['doc2']}
```

Expected out : list of sorted list with tuples.

```
example :[('a', ['doc2']), ('developer', ['doc2']), ('hello', ['doc1']), ('my', ['doc1']),
          ('name', ['doc1']), ('is', ['doc1', 'doc2']), ('james', ['doc1', 'doc2'])]
```

Note: You are allowed to use any built-in modules.