1.

1.1 What are the 5 essential characteristics that define Cloud Computing? Just state them.

1.2 Give an example of the characteristics you answered in 1.1, in action, in a hypothetical “reminder” messaging service. In this service, registered users create “reminders”, which are messages scheduled to be sent automatically by the service, at configurable times. Organize your answer by characteristic.

2. True (T) or False (F)?

2.1 As we transition from IaaS, to PaaS, to SaaS, what is abstracted increases, and more options become limited to the Provider's options.

2.2 In the short term, the option for Cloud Computing tends to be more expensive than the option for own resources.

2.3 Cloud Computing corresponds to the option for OpEx-oriented management, not CapitalEx-oriented.

2.4 A “Cloud Drive”, such as “DropBox”, “Google Drive”, etc., is a PaaS situation.

2.5 If a user loses important files in a “Cloud Drive”, because the “Cloud Drive” fails, the responsibility lies entirely with the Cloud Provider.

2.6 Isolation by deployment in a container is inferior to isolation by deployment in a virtual machine.

3.

Imagine yourself developing in AWS Cloud 9, on a newly created compute instance; i.e. for example Git (and other utilities) are available but not configured.

There is also a relevant, public remote repository, named “repo240517”, on github.com, owned by the user “cloud24”. The https URL of the repo is:

https://github.com/cloud24/repo240517

Indicate the git and/or bash commands needed to:

3.1. “Clone” the repository to the current folder

3.2. List existing branches.

3.3. Assume that there is only one branch, called “master”, which is your current context. How to list the files that are part of the most recent commit in master?

3.4. Create a new file 240517.txt, without content.

3.5. Add 240517.txt to the stage.

3.6. Is it possible to make a new commit? Yes or no? Justify.

3.7. Assume that a new commit was made with message “245017-1”, which includes the file “240517.txt”. Create and change the working context to a new branch named “new”.

3.8. Modify “240517.txt” and make a new commit with message “240517-2”.

3.9. Return to the master branch.

3.10. What is the content of “240517.txt” after the git checkout HEAD 240517.txt command?

4. In this question, the objective is to write a **Python Flask** app, to be Cloud deployed, that knows how to respond to GET requests with the following structure:

http://<domain name>:<port>/results/<year>/<month>

for example: http://localhost:5005/results/2024/5

Requests will originate from a very simple HTML interface, "**interface.html**" file, with content that you can read next, therefore, two numeric boxes, one to write a year, the other to write a month. This interface must be served on route /

There is also an "**interface.js**" file, the details of which are not relevant, nor will you have to write, but which ensures the invocation of the right URLs, depending on what is requested in the form.

```
<html><!-- interface.html -->
...
    <form id="idForm">
        <fieldset>
            <legend>Year and month:</legend>
                <label for="idYear">Year:</label>
                <input type="number" id="idYear"
placeholder="year" value="2024"><br>
                <label for="idMonth">Month:</label>
                <input type="number" id="idMonth"
placeholder="month" value="5">
            </fieldset>
        <input type="submit" value="get the results with
year/month">
    </form>
    <hr>
    <section id="idFeedback"></section>
...
</html>
```

Assume that this app (**app.py**) has access to a database of all results ever drawn in the "EuroMillions" gambling game.

The database is available in a text file "**DB.JSON**", in JSON format.

The file format is exactly the format of the responses that have to be produced, exemplified below.

The URLs http://<domain name>:<port>/results/<year>/<month> allow you to obtain text responses, structured in JSON. Each response is a JSON structure that represents the list of draws corresponding to the indicated year and month.

Below is an example of the response for 2020, month 2, limited to a single draw.

As noted, each response is a dictionary with keys **mDrawNumber**, **mYear**, **mMonth**, **mDay**, **mBalls** and **mStars**.

For this exercise, it is only important to understand that the **mYear** key represents the year of the draw; and the **mMonth** key is the month of the draw.

```
[

{"mDrawNumber": 1298, "mYear": 2020, "mMonth": 2, "mDay": 28,
"mBalls": [8, 11, 23, 20, 22], "mStars": [4, 3] },

…

]
```

4.1. Write a “tree” of directories and files, which clearly indicates the organization of your app, suitable for deployment in a Cloud environment capable of supporting your requirements.

4.2. Write the Python code necessary to operationalize the app. Do not use “list comprehension” techniques.

4.3. Indicate the necessary modifications (HTML and Python) so that the response is communicated to a **search_results.html** page, where each result appears as a member of an unordered list (ul).