Instructions:

- When writing python code be careful about spacing / indents. Leave enough room that it is obvious when you want things indented.
- You should pull off the first page of the exam to use as reference.
- Make sure to write legibly.
- Please write your name on the top of the next page.
- You will have one hour to complete the exam.
- You are allowed to use commands not covered in class, but if they do not work as expected on my computer they will not receive full credit.

Example Flask App

```
... # Previous code omitted

@app.route('/exam/1/<string:emp_name>', methods=['GET'])
def route_one(emp_name):
    emp_dept_info = {"alice": "engineering", "bob": "sales", "charlie": "engineering"}
    result_dict = {}
    if emp_name in emp_dept_info:
        result_dict[emp_name] = emp_dept_info[emp_name]
        return jsonify(result_dict), 200

    return jsonify({"error": "Employee not found"}), 404

@app.route('/exam/2/<string:emp_name>', methods=['GET'])
def route_two(emp_name):
    emp_info = workday_api_call(emp_name)
    # Returns a list of the form: [name (string), age (int), salary (float)]
    # Ex: ["Alice", 28, 98750.12]
    # If the employee does not exist it returns an empty list []

    # CODE GOES HERE

... # Remaining code omitted
```

Example Log File

```
2024-12-11 10:15:23 | ERROR | ERROR131 File Format
2024-12-11 10:15:28 | INFO | GET request received at /exam/2
2024-12-11 10:15:30 | INFO | GET request received at /exam/3
2024-12-11 10:15:35 | ERROR | GET request received at /exam/1
2024-12-11 10:16:01 | ERROR | ERROR131 File Format
2024-12-11 10:16:01 | ERROR | ERROR001 Database Connection
2024-12-11 10:16:02 | WARNING | Retrying database connection
2024-12-11 10:16:03 | INFO | Database connection restored
2024-12-11 10:16:05 | INFO | GET request received at /exam/2
```

```
/
├── Dockerfile
├── makefile
├── requirements.txt
├── src
│   └── app.py
├── logs
│   ├── 2024.12.07_flask.log
│   ├── 2024.12.08_flask.log
│   ├── 2024.12.09_flask.log
│   ├── 2024.12.10_flask.log
│   └── 2024.12.11_flask.log
└── data
    ├── docs
    │   ├── doc1.pdf
    │   ├── doc2.pdf
    │   ├── doc3.pdf
    │   └── doc4.pdf
    └── pdf
        └── result.pdf
```

```
FROM python:3.10.15-bookworm

WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

CMD ["python", "src/app.py"]
```

Makefile Example

```
IMAGE_NAME=flask_app

.PHONY=build interactive run

build:
        docker build . -t $(IMAGE_NAME)

interactive: build
        docker run -it -v $(shell pwd):/app/src $(IMAGE_NAME) /bin/bash

run: build
        docker run -p 4000:5000 -v $(shell pwd):/app/src $(IMAGE_NAME)
```

Name: _____

Group / Peer Review

Please list the names of your project group members and indicate what percentage of the work each person contributed. **The total percent should be roughly 100%**. If there are any comments you would like to state about your group dynamics, please use this space. You are welcome, but not required to add notes.

Member Name (include yourself)	Percentage of Work

Additional notes or comments:

1. (5 Points) Your current working directory is `/data/pdf` directory, using a *relative* path and a single command (do not change directories) please move all the PDF files in `/data/docs` to your current directory (`/data/pdf`).

2. Using `bash` commands please complete the following. You are currently in the `/logs` directory. The files contain logs generated by our custom logger, similar to what we demonstrated in class. Each file contains logs from the date specified in the file name. You can see an example of the format of the logs in the sample.

   (a) (5 Points) Create a file (`error_logs.log`) which contains all lines that have `ERROR` written in them (this should be *case sensitive*) from the file `2024.12.11_flask.log`. E.g. we want a file which contains only those lines.

   (b) (5 Points) Print to the screen the *number* of times `ERROR131` occurs in the log file `2024.12.11_flask.log`.

3. (8 Points) When we run `make interactive` what is the full path, *inside the container*, for the file `result.pdf`?

4. (5 Points) When we talk about testing, what does *coverage* mean? (1-2 sentences max)

5. Looking at our flask server code above, please write what is returned when the following calls are made. Please provide both the body and the status code.

   (a) (3 Points) A GET request to `/exam/1/bob`

   (b) (3 Points) A GET request to `/exam/1/dylan`

6. (2 Points) What does DRY stand for when speaking about code quality?

7. (10 Points) The code for `/exam/2` needs to be completed. The code should return:

   - If the employee exists it should return a JSON object of the form:
     {`"name": NAME OF PERSON, "age": AGE of PERSON`} with a status code of 200.
   - If the employee does not exist it should return an error message in JSON format (you can decide what it says) with a status code of 404.

8. (8 Points) Fill in the table below showing what CRUD operations stand for and their corresponding HTTP methods.

Letter	What it stands for	HTTP Method Used
C		
R		
U		
D		

9. (5 Points) Please write a unit test for the following function. This function only needs to have one test condition. Please name the function `test_math_func`.

```
def math_func(a, b, c):
    return (a * b) + c
```

10. (5 Points) In the Dockerfile shown above there is a line between the pip run and the final `CMD`. In that space, please write the command to set an environment variable (`API_KEY`) equal to `111222`.

11. (5 Points) Please write a function `g` which takes in three arguments: `odd_func`, `even_func` and `x`. `x` is an integer while `odd_func` and `even_func` are functions. If `x` is even it should return `even_func` with `x` provided as the first argument and if `x` is odd it should return `odd_func` with `x` provided as the first argument.