Task Overview:

You are tasked with building a client-server application using **ZeroMQ (ZMQ)** for communication between the client and server. The server will accept two types of commands: **OS commands** and **Math commands**. The client will send these commands in JSON format, and the server will process and return the result.

This task evaluates your ability to:

- Design a **well-structured** codebase.
- Use **design patterns** and best **coding practices**.
- Implement **efficient communication** using **ZMQ**.
- Handle different types of commands.

Deadline:

You have **4 days** to complete and submit the project.

Requirements:

1. Server:

- The server will process two types of commands:
  - **OS Commands** (e.g., Ping, list directories)
  - **Math Commands** (simple arithmetic expressions)
- The server should:
  - Listen for incoming JSON requests.
  - Identify the type of command (os or compute).
  - For **OS commands**: Execute the command on the system and return the result.
  - For **Math commands**: Evaluate the mathematical expression and return the result.
  - Handle exceptions (e.g., invalid commands or expressions).
  - Follow a **clean and modular design**. The server logic should be extensible (e.g., you can easily add new command types in the future).

2. Client:

- The client will:

- Accept a JSON structure representing either an OS or Math command.
- Send this command to the server using **ZMQ**.
- Display the server's response (command output or error).

3. Commands:

- The commands will follow the structure:

  Example 1: OS Command (Ping)

  json

```
{
  "command_type": "os",
  "command_name": "ping",
  "parameters": [
    "127.0.0.1",
    "-n",
    "6"
  ]
}
```

  Example 2: Math Command

  json

```
{
  "command_type": "compute",
  "expression": "(2 + 2) * 10"
}
```

Constraints:

- **OS Commands** should only be commands that can run on the operating system, like ping, ls, dir (depending on the OS).
- **Math Commands** should only support basic arithmetic operations such as addition, subtraction, multiplication, division, and parentheses.

Expected Deliverables:

1. A **well-structured project** that implements the client-server architecture.
2. Ensure that the server can handle multiple commands concurrently, i.e., it should be able to process multiple requests at once.
3. The code must be clean, well-commented, and follow consistent patterns.

Bonus:

- Implement command **logging** on the server side.
- Implement **unit tests** for the client and server.

Submit:

- Upload the project to GitHub (or provide a zip file).
- Include a **README.md** explaining how to run the project and a brief description of your design decisions.
- Email your **GitHub** project link to (m.rahimi@azmagroup.ir)

Evaluation Criteria:

1. **Code Structure**: How modular and scalable the solution is.
2. **Code Style**: Cleanliness, readability, and consistent naming conventions.
3. **ZMQ Usage**: Efficient use of ZMQ for client-server communication.
4. **Error Handling**: Robust handling of invalid or malformed commands.
5. **Concurrency**: Server's ability to handle multiple requests concurrently.
6. **Tests (Bonus)**: Presence of tests and code coverage.