API Theoretical Questions

1. What are the key things you would consider when creating/consuming an API to ensure that it is secure and reliable?

Ans: **Ensuring API Security**

- **Authentication & Authorization**: Use OAuth2, JWT, or API keys.
- **Data Encryption**: Enforce HTTPS for data in transit. Encrypt sensitive data at rest.
- **Input Validation**: Implement rigorous validation. Use whitelist approaches.
- **Rate Limiting**: Protect against brute-force and DDoS attacks.
- **Error Handling**: Avoid revealing sensitive information in errors.
- Dependency Management: Regularly update and audit dependencies.
- **Ensuring API Reliability**
- **Monitoring & Logging**: Monitor performance, error rates, and usage.
- **Rate Limiting & Throttling**: Manage load and ensure fair usage.
- **Caching**: Improve response times for frequent requests.
- **Load Balancing**: Distribute requests to enhance responsiveness.

CSV Theoretical Questions

1. How will you tackle the challenge above?

Ans:

- **Read the input line by line, splitting each entry into its key (cell identifier) and value (either a direct value or an expression).**
- **Store direct values immediately in a map to keep track of cell values.**
- **For expressions, store them for later evaluation to ensure all possible references can be resolved.**
- **To maintain the order of entries as they appear in the input, use HashMap for storing both values and expressions. This ensures the output matches the input order.**
- **After all direct values are stored, evaluate expressions. If an expression references another cell, ensure that cell's value is computed first. This might involve recursion, especially if that cell's value is also an expression.**

2. What type of errors you would you check for?

Ans:

- **Expressions referencing cells that do not exist in the input.**
- **Expressions that cannot be parsed or evaluated correctly (e.g., malformed formulas, unsupported operations).**
- **Non-integer Values: Input values or expressions that result in non-integer values where integers are expected.**
- **Incorrect formatting of cell identifiers or expressions (e.g., missing "=", invalid characters).**
- **Although not explicitly mentioned, if division operations were supported, this would be a critical check.**

3. How might a user break your code?

Ans:

- **Circular References: Inducing infinite recursion.**
- **Large Expressions: Causing performance issues or memory overflow.**
- **Malformed Inputs: Triggering parsing or runtime errors.**
- **Unsupported Operations: Introducing operations not handled by the code.**
- **Integer Overflow/Underflow: Generating incorrect results due to extreme values.**
- **Non-Integer Inputs: Providing inputs that cause type mismatches.**
- **Excessive Entries: Overwhelming the system's processing capabilities.**
- **Empty or Blank Entries: Causing unexpected behavior or errors.**
- **Special Characters: Leading to parsing errors or security vulnerabilities.**