LAB ASSIGNMENT-19.2

NAME :S RUCHITHA

ROLL NO:2403A52316

COURSE:AI ASSISTED CODING

BATCH:01(AIML)

QUESTIONS:

Lab Question 1: Sorting Algorithm Translation

You are part of a multinational development team. The backend is written in **Java**, but a new module requires a **Python** implementation of the same algorithm for integration with a data science pipeline.

- **Task 1:** Use AI-assisted coding to translate a given **Java bubble sort program** into Python. Verify that the translated code works correctly.
- **Task 2:** Introduce errors in the Python version to check if the input list is empty or contains non-numeric values.

Lab Question 2: File Handling Translation

A company's legacy codebases stores and processes files in **C++**, but the analytics team needs an equivalent program in **JavaScript (Node.js)** for integration with a web dashboard.

- **Task 1:** Translate a given C++ **file read-and-write program** into JavaScript using AI assistance. Ensure the script reads a text file and writes processed output to a new file.
- **Task 2:** Add error handling in the JavaScript version to gracefully handle missing files or permission errors.

Lab Question 3: API Call Translation

Your team developed a prototype in **Python** to fetch weather data from an API, but the production environment only supports **Java**.

- **Task 1:** Translate the Python script (that makes an API call and prints the response) into Java using AI-assisted coding. Ensure equivalent functionality.
- **Task 2:** Add proper error handling in the Java version for cases such as invalid API key, request timeout, or no internet connection.

LAB QUESTION 1:

PROMPT:

You are part of a multinational development team.
The backend system uses Java, but a new module requires a Python implementation for integration with a data science pipeline.

Task 1: Use AI-assisted coding to translate the following Java Bubble Sort program into Python and verify that it works correctly.

Task 2: Modify the Python version to include error handling that checks if the input list is empty or contains non-numeric values. If so, print an appropriate error message instead of sorting.
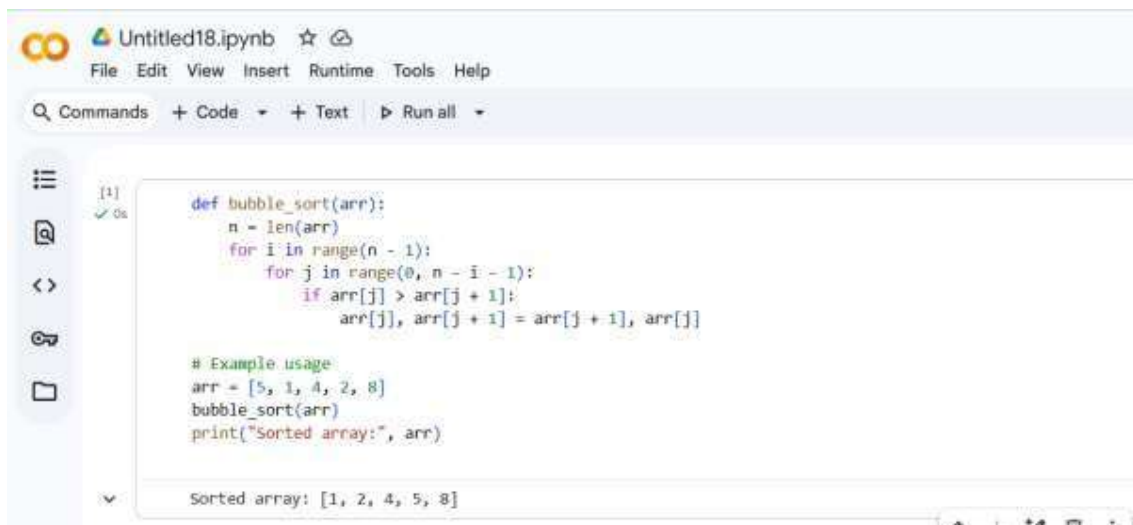
CODE:

```java
public class BubbleSort {
    public static void main(String[] args) {
int[] arr = {5, 1, 4, 2, 8};
bubbleSort(arr);
        System.out.println("Sorted array:");
for (int num : arr) {
            System.out.print(num + " ");
    }
  }

    static void bubbleSort(int[] arr) {
int n = arr.length;
```

```
    for (int i = 0; i < n - 1; i++) {
for (int j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {            int
temp = arr[j];          arr[j] =
arr[j + 1];          arr[j + 1] =
temp;
        }
      }
    }
  }
}
```
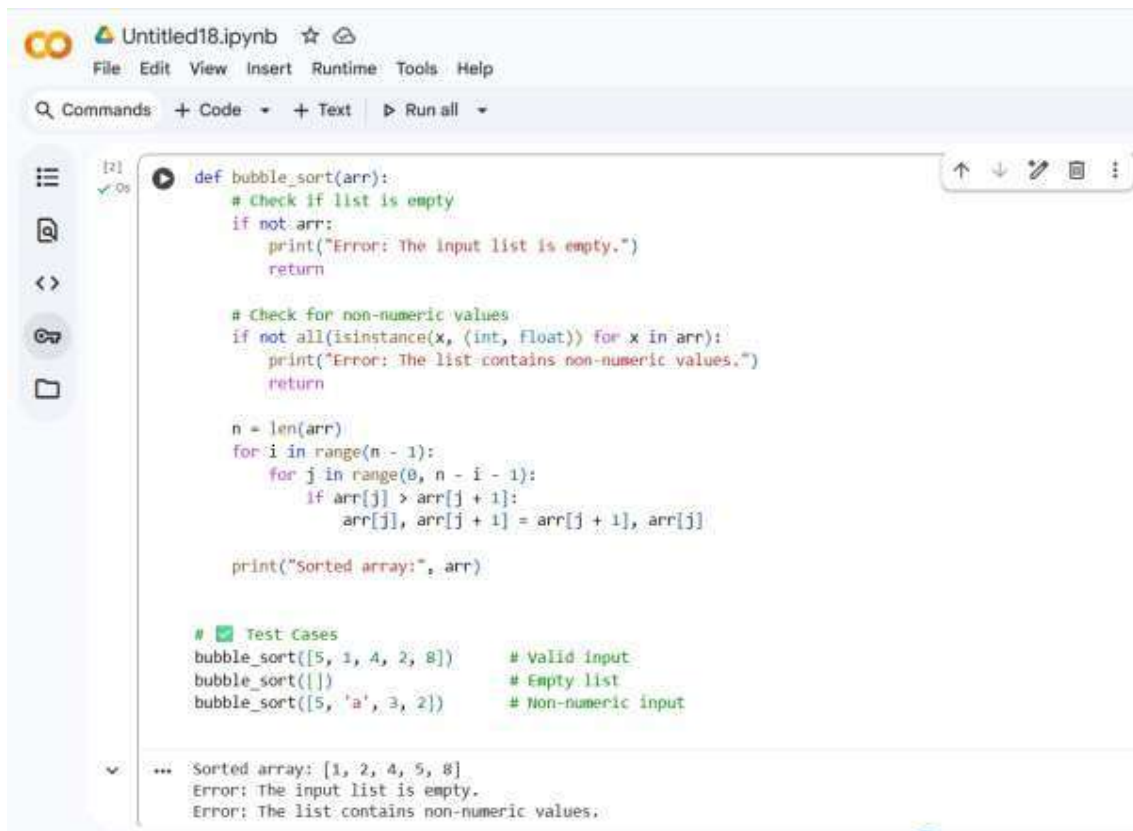


```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

# Example usage
arr = [5, 1, 4, 2, 8]
bubble_sort(arr)
print("Sorted array:", arr)
```

```
Sorted array: [1, 2, 4, 5, 8]
```

```python
def bubble_sort(arr):
    # Check if list is empty
    if not arr:
        print("Error: The input list is empty.")
        return

    # Check for non-numeric values
    if not all(isinstance(x, (int, float)) for x in arr):
        print("Error: The list contains non-numeric values.")
        return

    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

    print("Sorted array:", arr)


# ✅ Test Cases
bubble_sort([5, 1, 4, 2, 8])      # Valid input
bubble_sort([])                    # Empty list
bubble_sort([5, 'a', 3, 2])        # Non-numeric input
```

```
Sorted array: [1, 2, 4, 5, 8]
Error: The input list is empty.
Error: The list contains non-numeric values.
```

LAB QUESTION 2:

PROMPT:

You are part of a development team that is modernizing a company's legacy C++ system.
The old system reads a text file, processes its content, and writes the results to a new file.

Task 1: Use AI-assisted coding to translate the given C++ file read-and-write program into JavaScript (Node.js).
Ensure that the JavaScript version correctly reads data from a

text file and writes processed output (for example, uppercase text) into a new file.

Task 2

Extend the JavaScript (Node.js) file-handling program by adding **error handling**.
The updated script should handle missing input files, invalid permissions, or any I/O errors gracefully, showing userfriendly messages without crashing the program.

CODE :

```cpp
#include <iostream>

#include <fstream>

#include <string> using

namespace std;


int main() {
   ifstream inputFile("input.txt");
ofstream outputFile("output.txt");
string line;


   if (inputFile.is_open() && outputFile.is_open()) {
while (getline(inputFile, line)) {          outputFile
<< line << endl;
     }
     cout << "File has been copied successfully." << endl;
   } else {
     cout << "Error opening file." << endl;
   }


   inputFile.close();
   outputFile.close();
```

```
    return 0;

}
```



```
%%writefile file_processor.js

const fs = require('fs');

const inputFile = 'input.txt';
const outputFile = 'output.txt';

fs.readFile(inputFile, 'utf8', (err, data) => {
    if (err) {
        console.error(`Error reading file ${inputFile}:`, err);
        return;
    }

    // Process the content (e.g., convert to uppercase)
    const processedContent = data.toUpperCase();

    fs.writeFile(outputFile, processedContent, 'utf8', (err) => {
        if (err) {
            console.error(`Error writing file ${outputFile}:`, err);
            return;
        }
        console.log('File has been processed and copied successfully to output.txt.');
    });
});
```

Writing file_processor.js

```
///
```

✓0s ▾   Writing file_processor.js

Next, let's run the Node.js script we just created.

[6] ▷ !node file_processor.js
✓0s
▾   ••• File has been processed and copied successfully to output.txt.

Finally, let's verify the content of the `output.txt` file to ensure the translation and processing worked correctly.

[7]
✓0s
```python
with open('output.txt', 'r') as f:
    output_content = f.read()

print("Content of 'output.txt':\n---\n" + output_content + "\n---")
```

▾   Content of 'output.txt':

    HELLO FROM C++ LEGACY SYSTEM!
    THIS IS A TEST LINE.
    NODE.JS WILL PROCESS THIS.

---

[9] !mv input.txt input_original.txt
✓0s print("Renamed 'input.txt' to 'input_original.txt' to simulate a missing file.")

▾   Renamed 'input.txt' to 'input_original.txt' to simulate a missing file.

Now, let's run the Node.js script. It should now output an error message because `input.txt` is missing.

[10] !node file_processor.js
✓0s
▾   Error during reading of input.txt: File not found. Please ensure the input file exists.

As you can see, the script gracefully handled the missing file. Now, let's restore the original `input.txt` and run the script one more time to confirm it still works as expected when the file is present.

(+ Code) (+ Text)

[11] !mv input_original.txt input.txt
✓0s print("Restored 'input.txt'.")

▾   Restored 'input.txt'.

Running the script again with the `input.txt` file restored.

[12] !node file_processor.js
✓0s
▾   File has been processed and copied successfully to output.txt.

First, let's create a sample `input.txt` file that our Node.js script will read from.

First, let's create a sample `input.txt` file that our Node.js script will read from.

```python
with open('input.txt', 'w') as f:
    f.write('Hello from C++ legacy system!\n')
    f.write('This is a test line.\n')
    f.write('Node.js will process this.')

print("Created 'input.txt' successfully.")
```

```
Created 'input.txt' successfully.
```

Now, here is the JavaScript (Node.js) translation of your C++ program. It uses Node.js's built-in `fs` module to handle file operations. I've also added a step to convert the content to uppercase before writing it to `output.txt`.

```javascript
%%writefile file_processor.js

const fs = require('fs');

const inputFile = 'input.txt';
const outputFile = 'output.txt';

// Function to handle errors gracefully with more specific messages
function handleError(err, operation, filePath) {
    let errorMessage = `Error during ${operation} of ${filePath}: `;
    if (err.code === 'ENOENT') {
        errorMessage += 'File not found. Please ensure the input file exists.';
```

```javascript
// Function to handle errors gracefully with more specific messages
function handleError(err, operation, filePath) {
    let errorMessage = `Error during ${operation} of ${filePath}: `;
    if (err.code === 'ENOENT') {
        errorMessage += 'File not found. Please ensure the input file exists.';
    } else if (err.code === 'EACCES') {
        errorMessage += 'Permission denied. Please check file permissions.';
    } else {
        errorMessage += `An unexpected I/O error occurred: ${err.message}`;
    }
    console.error(errorMessage);
}

fs.readFile(inputFile, 'utf8', (err, data) => {
    if (err) {
        handleError(err, 'reading', inputFile);
        return;
    }

    // Process the content (e.g., convert to uppercase)
    const processedContent = data.toUpperCase();

    fs.writeFile(outputFile, processedContent, 'utf8', (err) => {
        if (err) {
            handleError(err, 'writing', outputFile);
            return;
        }
        console.log('File has been processed and copied successfully to output.txt.');
    });
});
```

LAB QUESTION 3:

PROMPT:

API Call Translation

You are part of a software development team migrating a data pipeline from Python to Java.
The Python prototype makes an API call to fetch weather data and prints the response.

Task 1: Use AI-assisted coding to translate the following Python API call program into Java while ensuring equivalent functionality (fetch data and print the JSON response).

Task 2

Extend the translated Java API program to include **robust error handling**.
The Java version should gracefully handle the following scenarios:

- Invalid API key (HTTP 401)

- Request timeout

- No internet connection or network issues

Print clear error messages for each case without the program crashing.

CODE:

```python
import requests


def get_weather():
```
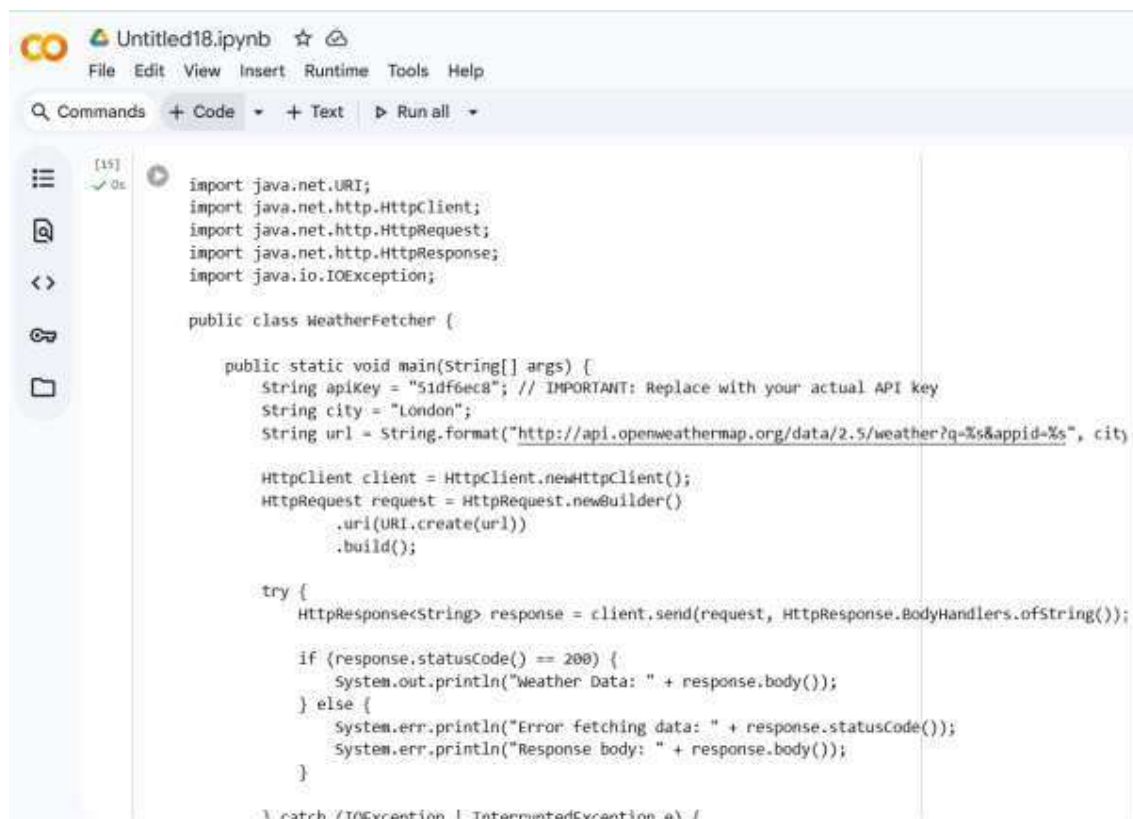
```python
    api_key = "your_api_key_here"
city = "London"    url =
f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}
"


    response = requests.get(url)
if response.status_code == 200:
    print("Weather Data:", response.json())
else:
    print("Error fetching data:", response.status_code)


get_weather()
```



```java
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.io.IOException;

public class WeatherFetcher {

    public static void main(String[] args) {
        String apiKey = "51df6ec8"; // IMPORTANT: Replace with your actual API key
        String city = "London";
        String url = String.format("http://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s", city

        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(url))
                .build();

        try {
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

            if (response.statusCode() == 200) {
                System.out.println("Weather Data: " + response.body());
            } else {
                System.err.println("Error fetching data: " + response.statusCode());
                System.err.println("Response body: " + response.body());
            }
        } catch (IOException | InterruptedException e) {
```

```
[15]     ....... ... - ...........(........ ..........................................................) ...
 ✓ 0s
                HttpClient client = HttpClient.newHttpClient();
                HttpRequest request = HttpRequest.newBuilder()
                        .uri(URI.create(url))
                        .build();

                try {
                    HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

                    if (response.statusCode() == 200) {
                        System.out.println("Weather Data: " + response.body());
                    } else {
                        System.err.println("Error fetching data: " + response.statusCode());
                        System.err.println("Response body: " + response.body());
                    }

                } catch (IOException | InterruptedException e) {
                    System.err.println("An error occurred during the API call: " + e.getMessage());
                    e.printStackTrace();
                }
            }
        }
```

Overwriting WeatherFetcher.java

Now, let's compile and run the Java code.

Since Colab primarily supports Python, we'll compile and run the Java code using shell commands. Please ensure

---

work. For demonstration purposes, I'll write this Java code to a file named `WeatherFetcher.java`.

```
[19]  ▷  %%writefile WeatherFetcher.java
 ✓ 0s
          import java.io.IOException;
          import java.net.URI;
          import java.net.http.HttpClient;
          import java.net.http.HttpRequest;
          import java.net.http.HttpResponse;
          import java.time.Duration; // Import Duration for timeouts

          public class WeatherFetcher {

              public static void main(String[] args) {
                  // IMPORTANT: Replace with your actual API key, or use an invalid one for testing 401 error
                  String apiKey = "51df6ec8";
                  String city = "London";
                  String url = String.format("http://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s", city);

                  HttpClient client = HttpClient.newBuilder()
                          .connectTimeout(Duration.ofSeconds(10)) // Set connection timeout
                          .build();

                  HttpRequest request = HttpRequest.newBuilder()
                          .uri(URI.create(url))
                          .timeout(Duration.ofSeconds(15)) // Set request timeout
                          .build();

                  try {
```

```java
try {
    HttpResponse<String> response = client.send(request, HttpResponse.Bod );

    if (response.statusCode() == 200) {
        System.out.println("Weather Data: " + response.body());
    } else if (response.statusCode() == 401) {
        System.err.println("Error: Invalid API Key. Please check your OpenWeatherMap API key.");
        System.err.println("Response body: " + response.body());
    } else if (response.statusCode() == 404) {
        System.err.println("Error: City not found or invalid request. Status Code: " + response
        System.err.println("Response body: " + response.body());
    } else {
        System.err.println("Error fetching data. Status Code: " + response.statusCode());
        System.err.println("Response body: " + response.body());
    }

} catch (IOException e) {
    if (e instanceof java.net.UnknownHostException) {
        System.err.println("Network Error: Unknown host. Check your internet connection or URL.
    } else if (e instanceof java.net.SocketTimeoutException) {
        System.err.println("Network Error: Request timed out. The server took too long to resp
    } else {
        System.err.println("Network or I/O Error: An error occurred during the API call: " + e.
    }
    e.printStackTrace();
} catch (InterruptedException e) {
    System.err.println("Error: The API call was interrupted: " + e.getMessage());
    Thread.currentThread().interrupt(); // Restore the interrupted status
    e.printStackTrace();
}
```