Ira A. Fulton Schools of Engineering
Arizona State University

CSE 531: Distributed and Multiprocessor Operating Systems

Logical Clock Project

Purpose

The goal of this project is to implement Lamport's logical clock algorithm upon the gRPC Project. As shown in Diagram A, the processes use logical clocks and follow Lamport's algorithm to coordinate.

Objectives

Learners will be able to:

- Implement the essential functions that enforces the client-centric consistency.
- Enforce the Monotonic Write policy, which extends the implementation of previous interfaces.
- Enforce the Read your Write policy, which extends the implementation of previous interfaces.
- Determine the problem statement.
- Identify the goal of the problem statement.
- List relevant technologies for the setup and their versions.
- Explain the implementation processes.
- Explain implementation results.
- implement Lamport’s logical clock algorithm.
- Implement interconnected sub-interfaces in the correct location.
- Execute the specific order of interconnected sub-interfaces.

Technology Requirements

- Access to Github
- Python
- gRPC
- You are required to use the files in the "Projects Overview and Resources" page in the *Welcome and Start Here* module of the course.

Directions

Part 1: Written Report

Your written report must be a single PDF with the correct naming convention: *CSE 531_Your Name_Logical Clock_Written Report*.

Using the provided *Learner Template_CSE 531_Your Name_Logical Clock_Written Report*, compose a report addressing the questions:

1. What is the problem statement and goal?
2. What is the goal of the problem statement?
3. What are the relevant technologies for the setup and their versions?
4. What are the implementation processes?
5. What are the results and their justifications?

*Learners may add subheadings on the template to purposefully call attention to specific, organized details.

Part 2: Project Code

Major Tasks

1. Implement logical clock in every customer and branch process
2. Implement Lamport's algorithm for clock coordination among the processes

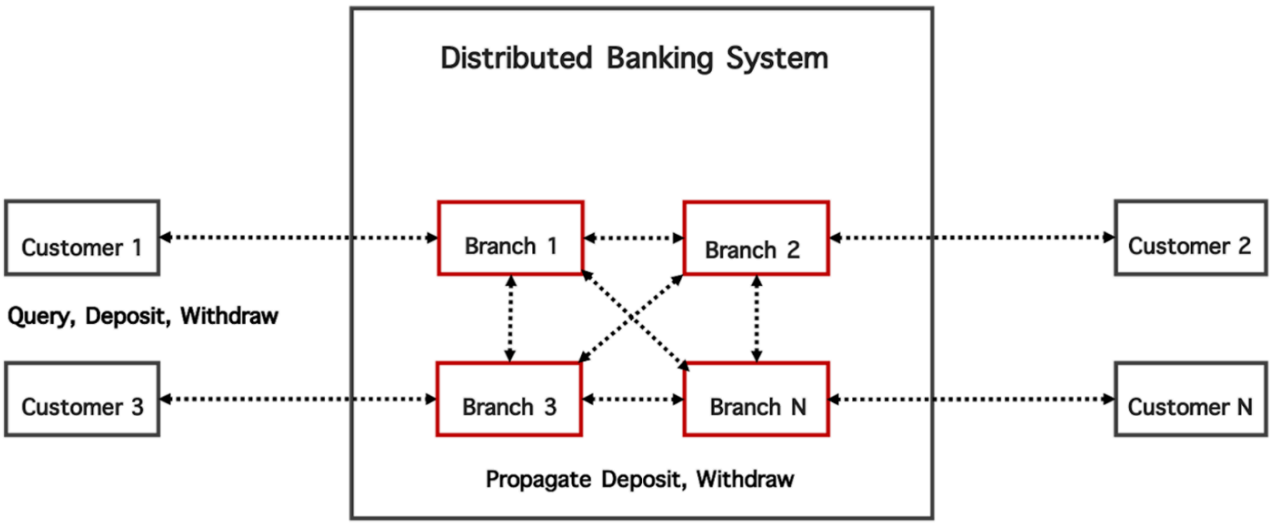

Diagram A: The processes use logical clocks and follow the Lamport's algorithm to coordinate

1. Description

Same as in the gRPC Project, customers use Query/Deposit/Withdraw interfaces to issue requests to branches, and branches use Propagate to propagate updates. Every customer Deposit/Withdraw request is handled by one branch which then propagates the update to the other branches.

As illustrated in **Diagram B**, every process maintains its logical lock according to the happens-before relationship between requests 1) of the same process and 2) between send and receive of the same request.

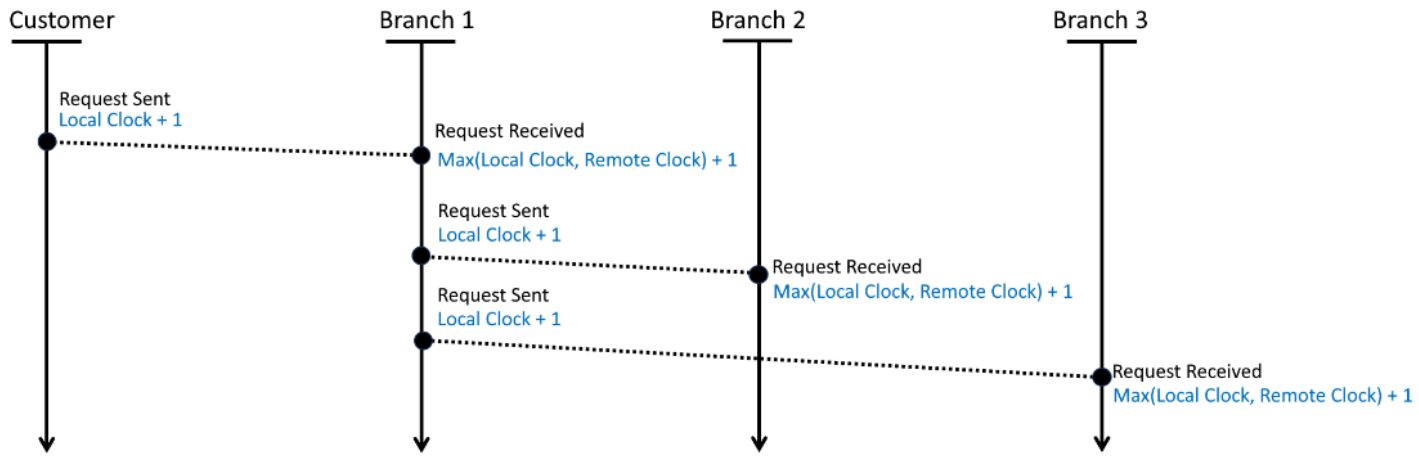

Diagram B: Events triggered by a customer request and their logical times

2. Input and Output

The input file contains a list of Customers and Branch processes. The format of the input file is similar to that of the gRPC Project. Note that the "customer-request-id" parameter in the "events" uniquely identifies every customer request, which needs to be carried over to all the events triggered by this request in order to verify the happens-before relationships are correctly enforced.

```
[ // Start of the Array of Branch and Customer processes
  { // Customer process #1
    "id" : {a unique identifier of a customer},
    "type" : "customer",
    " customer-requests " :  [{"customer-request-id": {unique identifier of a customer request} ,
"interface":{deposit | withdraw}, "money": {an integer value} }]
  }
```

```
 { … } // Customer process #2
 { … } // Customer process #3
 { … } // Customer process #N
 { // Branch process #1
   "id" : {a unique identifier of a branch},
   “type” : “branch”
   “balance” : {the initial amount of money stored in the branch}
 }
 { … } // Branch process #2
 { … } // Branch process #3
 { … } // Branch process #N
] // End of the Array of Branch and Customer processes
```

The output will contain three parts: (1) all the events (along with their logical times) taken place on each customer, (2) all the events (along with their logical times) taken place on each branch, and (3) all the events (along with their logical times) triggered by each customer Deposit/Withdraw request. For clarity, order all the events from the same customer/branch/customer-request by their logical times in the output.

```
[ // Start of the Customer processes
  { // Customer process #1
    “id” : {a unique identifier of a customer},
    “type”: “customer”,
    “events”: [{ “customer-request-id”: {unique identifier of the customer request}, “logical time”: {timestamp from the logical clock }} …]
  }
 { … } // Customer process #2
 { … } // Customer process #3
 { … } // Customer process #N
 // End of the Customer processes
```

```
 // Start of the of Branch processes
  { // Branch process #1
    “id” : {a unique identifier of a Branch},
    “type”: “branch”,
    “events”: [{ “customer-request-id”: {a unique identifier of the customer request that triggered this event}, “logical time”: { timestamp from the logical clock }} …]
  }
 { … } // Branch process #2
 { … } // Branch process #3
 { … } // Branch process #N
```

```
// End of the Branch processes
```

```
// Start of the events across all processes
  { // Event #1
    “customer-request-id” : {a unique identifier of a customer request}
    “logical time”: [{timestamp from the logical clock } …]
  }
 { … } // Event #2
 { … } // Event #3
 { … } // Event #N
] // End of the Events
```

Example of the Input File

```
[
   {
      "id": 1,
      "type": "customer",
      "customer-requests": [
         {
            " customer-request-id ": 1,
            "interface": "deposit",
            "money": 10
         },
         {
            " customer-request-id": 2,
            "interface": "withdraw",
            "money": 10
         }
      ]
   },
   {
      "id": 2,
      "type": "customer",
      "customer-requests": [
         {
            "customer-request-id ": 3,
            "interface": "deposit",
            "money": 10
         },
         {
            " customer-requests": 4,
```

```
            "interface": "withdraw",
            "money": 10
        }
    ]
  },
  {
    "id": 3,
    "type": "customer",
    "customer-requests": [
        {
            " customer-requests": 5,
            "interface": "deposit",
            "money": 10
        },
        {
            " customer-requests ": 6,
            "interface": "withdraw",
            "money": 10
        }
    ]
  },
  {
    "id": 1,
    "type": "branch",
    "balance": 400
  },
  {
    "id": 2,
    "type": "branch",
    "balance": 400
  },
  {
    "id": 3,
    "type": "branch",
    "balance": 400
  }
]
```

Expected Output File

```
// Part 1: List all the events taken place on each customer
[{
   "id":1,
```

```
   "type":"customer",
   "events": [
      {"customer-request-id": 1, "logical_clock": 1, "interface": "deposit", "comment": "event_sent from
customer 1"},
      {"customer-request-id": 2, "logical_clock": 2, "interface": "withdraw", "comment": "event_sent from
customer 1"}]
},
{
   "id":2,
   "type":"customer",
   "events": [
      {"customer-request-id": 3, "logical_clock": 1, "interface": "deposit", "comment": "event_sent from
customer 2"},
      {"customer-request-id": 4, "logical_clock": 2, "interface": "withdraw", "comment": "event_sent from
customer 2"}
   ]
},
{
   "id":3,
   "type":"customer",
   "events": [
      {"customer-request-id": 5, "logical_clock": 1, "interface": "deposit", "comment": "event_sent from
customer 3"},
      {"customer-request-id": 6, "logical_clock": 2, "interface": "withdraw", "comment": "event_sent from
customer 3"}]
}]

// Part 2: List all the events taken place on each branch
[{
   "id": 1,
   "type": "branch",
   "events":
   [{"customer-request-id": 1, "logical_clock": 2, "interface": "deposit", "comment": "event_recv from customer
1"},
   {"customer-request-id": 1, "logical_clock": 3, "interface": "propogate_deposit", "comment": "event_sent to
branch 2"},
   {"customer-request-id": 1, "logical_clock": 4, "interface": "propogate_deposit", "comment": "event_sent to
branch 3"},
   {"customer-request-id": 3, "logical_clock": 7, "interface": "propogate_deposit", "comment": "event_recv
from branch 2"},
   {"customer-request-id": 5, "logical_clock": 8, "interface": "propogate_deposit", "comment": "event_recv
from branch 3"},
```

```
   {"customer-request-id": 2, "logical_clock": 9, "interface": "withdraw", "comment": "event_recv from
customer 1"},
   {"customer-request-id": 2, "logical_clock": 10, "interface": "propogate_withdraw", "comment": "event_sent
to branch 2"},
   {"customer-request-id": 2, "logical_clock": 11, "interface": "propogate_withdraw", "comment": "event_sent
to branch 3"},
   {"customer-request-id": 4, "logical_clock": 14, "interface": "propogate_withdraw", "comment": "event_recv
from branch 2"},
   {"customer-request-id": 6, "logical_clock": 15, "interface": "propogate_withdraw", "comment": "event_recv
from branch 3"}]
},
{
   "id": 2,
   "type": "branch",
   "events":
   [{"customer-request-id": 1, "logical_clock": 4, "interface": "propogate_deposit", "comment": "event_recv
from branch 1"},
   {"customer-request-id": 3, "logical_clock": 5, "interface": "deposit", "comment": "event_recv from customer
2"},
   {"customer-request-id": 3, "logical_clock": 6, "interface": "propogate_deposit", "comment": "event_sent to
branch 1"},
   {"customer-request-id": 3, "logical_clock": 7, "interface": "propogate_deposit", "comment": "event_sent to
branch 3"},
   {"customer-request-id": 2, "logical_clock": 11, "interface": "propogate_withdraw", "comment": "event_recv
from branch 1"},
   {"customer-request-id": 4, "logical_clock": 12, "interface": "withdraw", "comment": "event_recv from
customer 2"},
   {"customer-request-id": 4, "logical_clock": 13, "interface": "propogate_withdraw", "comment": "event_sent
to branch 1"},
   {"customer-request-id": 5, "logical_clock": 14, "interface": "propogate_deposit", "comment": "event_recv
from branch 3"},
   {"customer-request-id": 4, "logical_clock": 15, "interface": "propogate_withdraw", "comment": "event_sent
to branch 3"},
   {"customer-request-id": 6, "logical_clock": 18, "interface": "propogate_withdraw", "comment": "event_recv
from branch 3"}]
},
{
   "id": 3,
   "type": "branch",
   "events":
   [{"customer-request-id": 5, "logical_clock": 2, "interface": "deposit", "comment": "event_recv from customer
3"},
```

{"customer-request-id": 5, "logical_clock": 3, "interface": "propogate_deposit", "comment": "event_sent to branch 1"},
{"customer-request-id": 1, "logical_clock": 5, "interface": "propogate_deposit", "comment": "event_recv from branch 1"},
{"customer-request-id": 5, "logical_clock": 6, "interface": "propogate_deposit", "comment": "event_sent to branch 2"},
{"customer-request-id": 3, "logical_clock": 8, "interface": "propogate_deposit", "comment": "event_recv from branch 2"},
{"customer-request-id": 2, "logical_clock": 12, "interface": "propogate_withdraw", "comment": "event_recv from branch 1"},
{"customer-request-id": 6, "logical_clock": 13, "interface": "withdraw", "comment": "event_recv from customer 3"},
{"customer-request-id": 6, "logical_clock": 14, "interface": "propogate_withdraw", "comment": "event_sent to branch 1"},
{"customer-request-id": 4, "logical_clock": 16, "interface": "propogate_withdraw", "comment": "event_recv from branch 2"},
{"customer-request-id": 6, "logical_clock": 17, "interface": "propogate_withdraw", "comment": "event_sent to branch 2"}]

}]

// Part 3: List all the events (along with their logical times) triggered by each customer Deposit/Withdraw request

[{"id": 1,"customer-request-id":1,"type": "customer","logical_clock": 1,"interface": "deposit","comment": "event_sent from customer 1"},

{"id": 1,"customer-request-id":1,"type": "branch","logical_clock": 2,"interface": "deposit","comment": "event_recv from customer 1"},

{"id": 1,"customer-request-id":1,"type": "branch","logical_clock": 3,"interface": "propogate_deposit","comment": "event_sent to branch 2"},

{"id": 2,"customer-request-id":1,"type": "branch","logical_clock": 4,"interface": "propogate_deposit","comment": "event_recv from branch 1"},

{"id": 1,"customer-request-id":1,"type": "branch","logical_clock": 4,"interface": "propogate_deposit","comment": "event_sent to branch 3"},

{"id": 3,"customer-request-id":1,"type": "branch","logical_clock": 5,"interface": "propogate_deposit","comment": "event_recv from branch 1"},

{"id": 1,"customer-request-id":2,"type": "customer","logical_clock": 2,"interface": "withdraw","comment": "event_sent from customer 1"},

{"id": 1,"customer-request-id":2,"type": "branch","logical_clock": 9,"interface": "withdraw","comment": "event_recv from customer 1"},

{"id": 1,"customer-request-id":2,"type": "branch","logical_clock": 10,"interface": "propogate_withdraw","comment": "event_sent to branch 2"},

{"id": 2,"customer-request-id":2,"type": "branch","logical_clock": 11,"interface": "propogate_withdraw","comment": "event_recv from branch 1"},

{"id": 1,"customer-request-id":2,"type": "branch","logical_clock": 11,"interface": "propogate_withdraw","comment": "event_sent to branch 3"},

{"id": 3,"customer-request-id":2,"type": "branch","logical_clock": 12,"interface": "propogate_withdraw","comment": "event_recv from branch 1"},

{"id": 2,"customer-request-id":3,"type": "customer","logical_clock": 1,"interface": "deposit","comment": "event_sent from customer 2"},

{"id": 2,"customer-request-id":3,"type": "branch","logical_clock": 5,"interface": "deposit","comment": "event_recv from customer 2"},

{"id": 2,"customer-request-id":3,"type": "branch","logical_clock": 6,"interface": "propogate_deposit","comment": "event_sent to branch 1"},

{"id": 2,"customer-request-id":3,"type": "branch","logical_clock": 7,"interface": "propogate_deposit","comment": "event_sent to branch 3"},

{"id": 1,"customer-request-id":3,"type": "branch","logical_clock": 7,"interface": "propogate_deposit","comment": "event_recv from branch 2"},

{"id": 3,"customer-request-id":3,"type": "branch","logical_clock": 8,"interface": "propogate_deposit","comment": "event_recv from branch 2"},

{"id": 2,"customer-request-id":4,"type": "customer","logical_clock": 2,"interface": "withdraw","comment": "event_sent from customer 2"},

{"id": 2,"customer-request-id":4,"type": "branch","logical_clock": 12,"interface": "withdraw","comment": "event_recv from customer 2"},

{"id": 2,"customer-request-id":4,"type": "branch","logical_clock": 13,"interface": "propogate_withdraw","comment": "event_sent to branch 1"},

{"id": 1,"customer-request-id":4,"type": "branch","logical_clock": 14,"interface": "propogate_withdraw","comment": "event_recv from branch 2"},

{"id": 2,"customer-request-id":4,"type": "branch","logical_clock": 15,"interface": "propogate_withdraw","comment": "event_sent to branch 3"},

{"id": 3,"customer-request-id":4,"type": "branch","logical_clock": 16,"interface": "propogate_withdraw","comment": "event_recv from branch 2"},

{"id": 3,"customer-request-id":5,"type": "customer","logical_clock": 1,"interface": "deposit","comment": "event_sent from customer 3"},

{"id": 3,"customer-request-id":5,"type": "branch","logical_clock": 2,"interface": "deposit","comment": "event_recv from customer 3"},

{"id": 3,"customer-request-id":5,"type": "branch","logical_clock": 3,"interface": "propogate_deposit","comment": "event_sent to branch 1"},

{"id": 3,"customer-request-id":5,"type": "branch","logical_clock": 6,"interface": "propogate_deposit","comment": "event_sent to branch 2"},

{"id": 1,"customer-request-id":5,"type": "branch","logical_clock": 8,"interface": "propogate_deposit","comment": "event_recv from branch 3"},

{"id": 2,"customer-request-id":5,"type": "branch","logical_clock": 14,"interface": "propogate_deposit","comment": "event_recv from branch 3"},

{"id": 3,"customer-request-id":6,"type": "customer","logical_clock": 2,"interface": "withdraw","comment": "event_sent from customer 3"},

{"id": 3,"customer-request-id":6,"type": "branch","logical_clock": 13,"interface": "withdraw","comment": "event_recv from customer 3"},

{"id": 3,"customer-request-id":6,"type": "branch","logical_clock": 14,"interface": "propogate_withdraw","comment": "event_sent to branch 1"},

{"id": 1,"customer-request-id":6,"type": "branch","logical_clock": 15,"interface": "propogate_withdraw","comment": "event_recv from branch 3"},

{"id": 3,"customer-request-id":6,"type": "branch","logical_clock": 17,"interface": "propogate_withdraw","comment": "event_sent to branch 2"},

{"id": 2,"customer-request-id":6,"type": "branch","logical_clock": 18,"interface": "propogate_withdraw","comment": "event_recv from branch 3"}]

Formatting Specifications

Naming convention for your project files and usage with the input file:

- The server file should be named server.py, all required servers should be able to start with this command **python server.py input.json** (execution command to start servers)

- Client file should be named client.py, client file should execute with **python client.py input.json**.

Protobuf: proto file should be inside the protos folder (e.g., **protos/banks.proto**). The expected file structure (excludes files generated after running proto file) should be like this:

```
├──protos
├── ├──banks.proto
├──input.json
├──server.py
├──client.py
├──customer.py
├──branch.py
├──output.json
```

Submission Directions for Deliverables

Part 1: Written Report

You are given a limited number of attempts to submit your best work. The number of attempts is given to anticipate any submission errors you may have in regards to properly submitting your best work within the deadline (e.g., accidentally submitting the wrong paper). It is not meant for you to receive multiple rounds of feedback and then one (1) final submission. Only your most recent submission will be assessed.

You must submit your Logical Clock Project Written Report deliverable in its submission space in the course. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

The Logical Clock Project Written Report includes one (1) deliverable:

- **Written Report:** Your written report must be a single PDF with the correct naming convention: *CSE 531_Your Name_Logical Clock_Written Report.*

Part 2: Project Code

You are given an unlimited number of attempts to submit your best work. You must submit your Logical Clock Project Code deliverable through Gradescope. Carefully review submission directions outlined in this overview document in order to correctly earn credit for your work. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

Submitting to Gradescope

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from Gradescope into your Canvas grade.

1. Go to the Canvas Assignment, "**Submission: Logical Clock Project Code**".
2. Click the "**Load Submission…in new window**" button.
3. Once in Gradescope, select the project titled, "**Submission: Logical Clock Project Code**", and a pop-up window will appear.
4. In the pop-up,
    a. Submit a single ZIP file.
    b. Click "**Upload**" to submit your work for grading.
5. You will know you have completed the assignment when feedback appears for each test case with a score.
6. If needed: to resubmit the assignment in Gradescope:
    a. Click the "**Resubmit**" button on the bottom right corner of the page and repeat the process from Step 3.

The Logical Clock Project Code includes one (1) deliverable:

- **ZIP File:** Your ZIP file must contain your Logical Clock Project code files and final output. The **code files** must follow the naming conventions outlined in the "Formatting Specifications" section of this Overview Document. The **final output** must be a single JSON file with the correct naming convention: **output.json**.
    - Zip your files by selecting all of them together (your code files should be in the root folder).

Evaluation

Project deliverables will be evaluated based on criteria and will receive a total score.

Evaluation details vary depending on whether the component is auto-graded or course team-graded, so review this section carefully so you understand how you earn credit for each portion of your work.

Review the course syllabus for details regarding late penalties.

Test Case

This component of the project is **auto-graded** and worth **40%** of your project grade.

Each customer and branch has their own logical clock, and it is updated during various events. The codes handle the propagation of 'deposits' and 'withdraws' events happening between customers and the branches. When an event is sent from one branch or customer to another, the clock is incremented and updated accordingly. The code logs events with details. The program outputs a json file that should show that these events should be propagated and well-ordered among different branches in the system, the logical clock is increasing for each event sent from the customer.

Your output file will be checked that the order of the events is correct. All the events should follow a happens-before relationship. Please refer to the sample **Expected Output File** in the "**Input and Output**" section of Part 2: Project Code.

Your grade for this portion will be assigned based on the **percentage** of returned values that follow the **correct order**. This schema for calculating the grade is also provided in the **rubric**:

- About **20%** of the events follow the correct order.
- About **40%** of the events follow the correct order.
- About **60%**of the events follow the correct order.
- About **80%** of the events follow the correct order.
- **100%** of the events follow the correct order.

Code

This component of the project is **course team-graded** and worth **40%** of your project grade.

Review the **rubric** for how your code will be graded.

Project deliverables missing any part of the project will be graded based on what was submitted against the rubric criteria. Missing parts submitted after the deadline will not be graded.

Report

This component of the project is **course team-graded** and worth **20%** of your project grade.

Review the **rubric** for how your written report will be graded.

Project deliverables missing any part of the project will be graded based on what was submitted against the rubric criteria. Missing parts submitted after the deadline will not be graded.

Rubric

Rubrics communicate specific criteria for evaluation. Prior to starting any graded coursework, learners are expected to read through the rubric so they know how they will be assessed. You are encouraged to self-assess your responses and make informed revisions before submitting your final report. Engaging in this learning practice will support you in developing your best work.

Component	Criteria	No Attempt	Undeveloped	Developing	Approaching	Proficient	Exemplary
Test Case	Check the order of the events is correct. All the events should follow a happens-before relationship.	Provided no response.	About 20% of the events follow the correct order.	About 40% of the events follow the correct order.	About 60% of the events follow the correct order.	About 80% of the events follow the correct order.	All of the events follow the correct order.
Component	**Criteria**	**No Attempt**	**Undeveloped**	**Developing**	**Approaching**	**Proficient**	**Exemplary**
Code	The code consists of the same components as the gRPC Project (list provided below), implementation of logical clock for both customers and branches, and implementation of the logical clock algorithm. Components: Branch.Query, Branch.Withdraw, Branch.Deposit, Branch.Propogate_Withdraw, Branch.Propagate_Deposit	Provided no response.	Provided project code that is syntactically or semantically invalid.	Provided project code is functional. Project code performs a few of the functions as described in the final report. Project code is not engineered or designed. No document-ation is provided.	Provided project code is functional. Project code performs most of the functions as described in the final report. Project code is thought-out and engineered. Project code provides documentation and code comments (where appropriate).	Provided project code is functional. Project code performs most of the functions as described in the final report. Project code is thought-out and engineered. Project code provides documentation and code comments (where appropriate).	Provided project code that is functional. Project code performs the functions as described in the final report. Project code is excellently engineered. Project code provided helpful documentation and code comments (where appropriate).
Component	**Criteria**	**No Attempt**	**Undeveloped**	**Developing**	**Approaching**	**Proficient**	**Exemplary**
Report	Problem statement and goal	Provided no response.	Provided an incoherent problem statement and goal with no connection to the project description.	Provided a somewhat coherent problem statement and Goal with little or misguided connections with the project description.	Provided a basic, understandable problem statement and goal that loosely connects with the project description.	Provided a clear problem statement and goal that directly connects with the project description.	Provided a focused problem statement and goal that distinctly and in meaningful ways connects with the project description.

	Setup	Provided no response.	Provided an incomplete explanation of the setup.	Provided a reasonable explanation of the setup, but missing steps.	Provided a complete explanation of the setup, but some steps are not working.	Provided complete and understandable explanation of the setup. All steps are working.	Provided complete and clear explanation of the setup. All steps are accurate and working correctly.
	Implementation Processes	Provided no response.	Provided an incomplete explanation and/or one or more of the major tasks may be missing.	Provided a general explanation and implementation . Inaccuracies may be present.	Provided a reasonable explanation and implementation . Steps may be illogical or missing.	Provided a logical explanation and implementation . Steps are logical, but may be disjointed or unrelated to one another.	Provided a sound explanation and implementation . All steps are accurate, related, and working correctly.
	Results	Provided no response.	Provided incorrect or fake results. The explanation is fully incorrect with no merit in approach or connection with the results.	Provides some results and/or unrelated results. The explanation is somewhat incorrect with little merit in approach or connection with the results.	Provides mostly correct results. The explanation is reasonable with some ambiguity.	Results are accurate. The explanation is logical with no ambiguity.	Results are accurate. The explanation is well-developed with strong justifications directly related to the implementation results.