# CPSC 474-01 Project 01 (Lamport's Logical Clocks) Report

Date: 2021-09-26

Written for: Professor Doina Bein, CSU Fullerton

Group Member(s): Kenneth Doan

E-Mail Address(es): snarbolax@csu.fullerton.edu

CPSC 474 Section 01

## Report Summary

The first part of the report contains the pseudocode of *algorithm_calculate()*--which calculates the logical clock values of events in an event matrix--and *algorithm_verify()*--which determines if a given LC matrix has only correct executions and will create an event matrix appropriate for the given LC matrix.

The second section of the report details how to run the program. Additional information about the program--including any additional execution notes--may be found in the included **README.md** file. The **README.md** file should be previewable and displayed on the repository's main page.

The final portion of the report includes the snapshot of group member(s) and names of the text files that contain the program's code snapshots; the specified text files should be located in the same directory as this report on the repository.

## Pseudocode

```
STRUCT process
    INIT process_count
    INIT event_count
    INIT events with 25 elements of "NULL"
    INIT LC_values with 25 elements of "0"
END STRUCT

STRUCT send_event
    INIT send_num
    INIT LC_value
END STRUCT
```

### algorithm_calculate()

```
FOR each processor in the matrix
    FOR each event in the processor
        INIT char_pointer to first character of event
        IF is the first event THEN
            IF char_pointer is 's' THEN
                SET char_pointer to second character of event
                SET LC value of event in processor to 1
                INIT accounted_event to false
                FOR each element in send_list
                    IF char_pointer equals to element's send_num THEN
                SET accounted_event to true
            END IF
        END FOR
        IF accounted_event is false THEN
            INIT temp_send as an empty send_event
            SET temp_send’s send_num to second char of event
            SET temp_send’s LC value to 1
            SET temp_send to element in send_list
            INCREMENT send_list_count
            CALL algorithm_calculate with updated LC values
        END IF
    ELSE-IF char_pointer is ‘r’ THEN
        SET char_pointer to second character of event
        FOR each element in send_list
            IF char_pointer equals to element’s send_num THEN
                SET LC value of event to element’s LC value+1
            END IF
        END FOR
    ELSE-IF char_pointer is alpha-numeric
        SET char_pointer to first character of event
        IF char_pointer is ‘N’ THEN
            FOR remaining number of events in proc_list
                SET LC value of event to 0
            END FOR
        ELSE-IF char_pointer is not ‘N’ THEN
            SET LC value of event to 1
        END IF
    END IF
ELSE is not first event THEN
    IF char_pointer is ‘s’ THEN
        SET char_pointer to second character of event
        SET LC value of event to preceding event’s LC value+1
        INIT accounted_event to false
        FOR each element in send_list
            IF char_pointer equals to element’s send_num THEN
                IF element value doesn’t match value in proc_list THEN
                    SET LC value of element to proc_list value+1
                    CALL algorithm_calculate
                END IF
                SET accounted_event to true
            END IF
        END FOR
        IF accounted_event is false THEN
            INIT temp_send as an empty send_event
```

```
                    SET temp_send's send_num to second char of event
                    SET temp_send's LC value to preceding event value+1
                    SET temp_send to element in send_list
                    INCREMENT send_list_count
                    CALL algorithm_calculate with updated LC values
                END IF
            ELSE-IF char_pointer is 'r' THEN
                SET char_pointer to second character of event
                FOR each element in send_list
                    IF char_pointer equals to element's send_num THEN
                        SET element value max+1 between send_list / proc_list
                    END IF
                END FOR
            ELSE-IF char_pointer is alpha-numeric
                IF char_pointer is 'N' THEN
                    FOR remaining number of events in proc_list
                        SET LC value of event to 0
                    END FOR
                ELSE-IF char_pointer is not 'N' THEN
                    SET LC value of event to preceding event value+1
                END IF
            END IF
        END IF
    END FOR
END FOR
```

## algorithm_verify()

```
INIT internal_vector
INIT receive_vector
INIT send_vector
INIT accounted_event

FOR each event in matrix
    INIT curr_column
    FOR each processor in matrix
        INIT curr_row
        INIT value to event's LC value
        CALL push_back on curr_row with value
        IF value is not 0 THEN
            IF value is 1 and event is not in accounted_event THEN
                CALL push_back on accounted_event with value
            END IF
            IF value is not 1 and event preceding in value is not in accounted_event THEN
```

```
                FOR each event in process
                    INIT checking to event's LC value
                    CALL push_back on curr_row with checking
                END FOR
                IF is first event THEN
                    CALL push_back on receive_vector with value
                    CALL push_back on send_vector with value-1
                    IF event is not in accounted_event THEN
                        CALL push_back on accounted_event with value
                    END IF
                ELSE-IF is not first event THEN
                    IF value previous event in curr_row not equal to value-1 THEN
                        CALL push_back on receive_vector with value
                        CALL push_back on send_vector with value-1
                        IF event is not in acknowledged_event THEN
                            CALL push_back on acced_event with value
                        END IF
                    ELSE
                        CALL push_back on internal_vector with value
                        IF event is not in acknowledged_event THEN
                            CALL push_back on acced_event with value
                        END IF
                    END IF
                END IF
            ELSE
                CALL push_back on internal_vector with value
                IF event is not in acknowledged_event THEN
                    CALL push_back on accounted_event with value
                END IF
            END IF
        END IF
    END FOR
END FOR
DETERMINE execution correctness RETURNING correct
IF correct equals false THEN
    PRINT "INCORRECT"
ELSE-IF correct equals true THEN
    COMPUTE event matrix
END IF

RETURN correct
```

## How to Run the Program

There are 2 primary methods of executing the program.

1. Double-click on **project01.exe**
2. Navigate to the directory that contains **project01.exe** with your OS's terminal / command-line.
    a. Type **project01.exe** into the terminal / command-line and press enter.

If you are executing the program via terminal / command-line, you may specify a particular output file to use and collect the program's output.

- You can specify a different output file to use by including the name of the output file--extension included--as the second parameter of the program execution line.
    - the output file has to be in the same directory of the program
    - e.g. "**project01.exe insert_user_custom_output_file_here.txt**", without the quotation marks
    - you are still able to specify a different output file if you execute the program by double-clicking on it

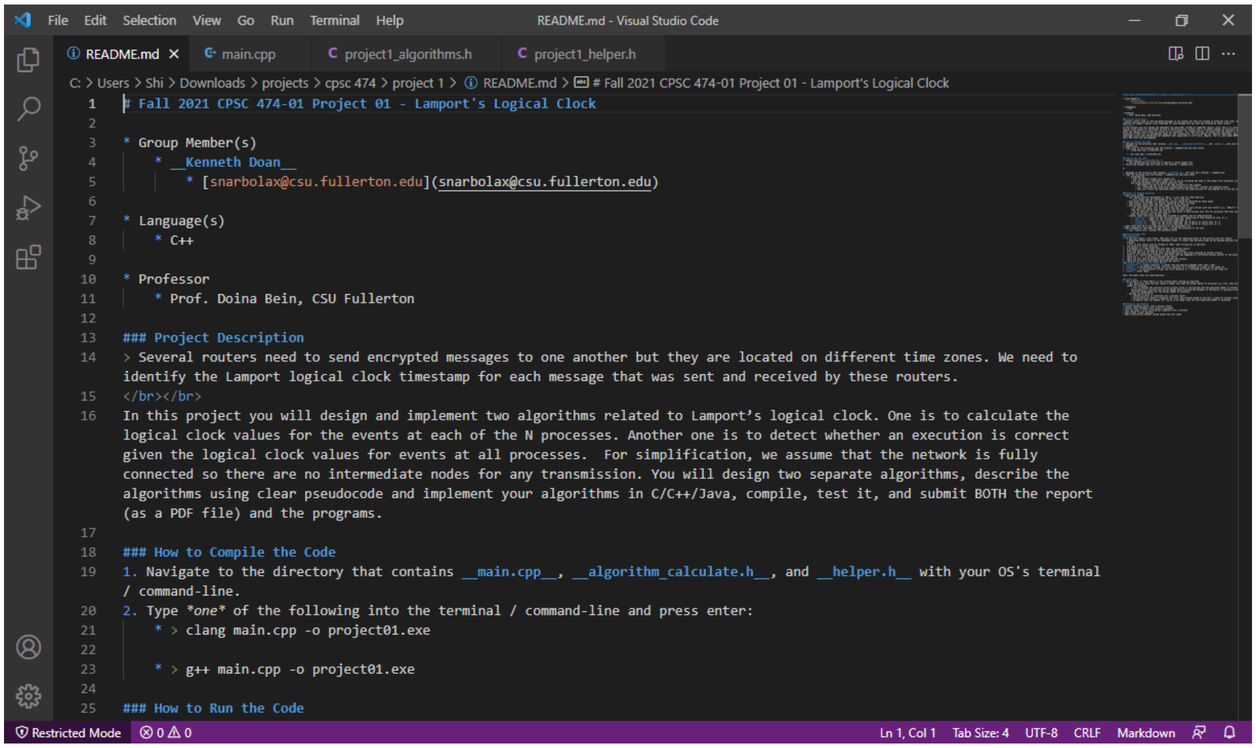
Otherwise, both primary methods of executing the program will use **output.txt** as the default output file.

If the user attempts to use an output file that does not exist--(in the same directory as the program)--as the non-default output file, a file with the specified name will be created.

While running the program, if the program prompts the user for the name of an input file, the name of the input file must be in the same directory as the program and has to include its file extension along with its name (e.g. "**user_custom_input_file.txt**").

## Snapshots

### Group Member(s)


File Edit Selection View Go Run Terminal Help
README.md - Visual Studio Code
README.md
main.cpp
project1_algorithms.h
project1_helper.h
C: > Users > Shi > Downloads > projects > cpsc 474 > project 1 > README.md > # Fall 2021 CPSC 474-01 Project 01 - Lamport's Logical Clock
Restricted Mode
0 0
Ln 1, Col 1 Tab Size: 4 UTF-8 CRLF Markdown

```
# Fall 2021 CPSC 474-01 Project 01 - Lamport's Logical Clock

* Group Member(s)
    * __Kenneth Doan__
        * [snarbolax@csu.fullerton.edu](snarbolax@csu.fullerton.edu)

* Language(s)
    * C++

* Professor
    * Prof. Doina Bein, CSU Fullerton

### Project Description
> Several routers need to send encrypted messages to one another but they are located on different time zones. We need to
identify the Lamport logical clock timestamp for each message that was sent and received by these routers.
</br></br>
In this project you will design and implement two algorithms related to Lamport's logical clock. One is to calculate the
logical clock values for the events at each of the N processes. Another one is to detect whether an execution is correct
given the logical clock values for events at all processes.  For simplification, we assume that the network is fully
connected so there are no intermediate nodes for any transmission. You will design two separate algorithms, describe the
algorithms using clear pseudocode and implement your algorithms in C/C++/Java, compile, test it, and submit BOTH the report
(as a PDF file) and the programs.

### How to Compile the Code
1. Navigate to the directory that contains __main.cpp__, __algorithm_calculate.h__, and __helper.h__ with your OS's terminal
/ command-line.
2. Type *one* of the following into the terminal / command-line and press enter:
    * > clang main.cpp -o project01.exe

    * > g++ main.cpp -o project01.exe

### How to Run the Code
```

### Code I/O

Included as:

- **calc1_output.txt**
    - Output file of *calc1.txt*, N = 3
- **calc2_output.txt**
    - Output file of *calc2.txt*, N = 5
- **verify1_output.txt**
    - Output file of *verify1.txt*, N = 3
- **verify2_output.txt**
    - Output file of *verify2.txt*, N = 5