CPSC 474-01 Project 02 (Programming using MPI) Report

Date: 2021-11-18

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

Franklin's Algorithm

```
CALL srand with time and rank

IF rank is not '0' THEN
        SET status to rand() % 2
END IF
IF status is '1' THEN
        PRINT process is initiator
END IF

WHILE global leader is '-1'
        CALL MPI_Barrier with MPI_COMP_WORLD
        IF status is '0' THEN
                IF rank is '0' THEN
                        CALL MPI_Recv with  size - 1 RETURNING process's left buffer
                ELSE
                        CALL MPI_Recv with (rank - 1) % size RETURNING process's left buffer
                END IF
                CALL MPI_Send with process's left buffer TO process's right side
                CALL MPI_Recv with  (rank + 1) % size RETURNING process's right buffer
                IF rank is '0' THEN
                        CALL MPI_Send with process's right buffer TO process's left side
```

```
            ELSE
                CALL MPI_Send with process's right buffer TO process's left side
            END IF
        ELSE IF status is '1' THEN
            IF rank is '0' THEN
                CALL MPI_Send with process's rank buffer TO process's left side
            ELSE
                CALL MPI_Send with process's rank buffer TO process's left side
            END IF
            CALL MPI_Send with process's rank buffer TO process's right side
            IF rank is '0' THEN
                CALL MPI_Recv with size - 1 RETURNING process's left buffer
            ELSE
                CALL MPI_Recv with (rank - 1) % size RETURNING process's left buffer
            END IF
            CALL MPI_Recv with (rank + 1) % size RETURNING process's right buffer

            IF left buffer is GREATER THAN right buffer
                SET max to left buffer
            ELSE IF left buffer is EQUAL TO right buffer
                SET max to left buffer
            ELSE IF left buffer is LESS THAN right buffer THEN
                SET max to right buffer
            END IF
            PRINT left buffer and right buffer

            IF max is LESS THAN rank THEN
                PRINT  rank is more than max
                INCREMENT round
            ELSE IF max is GREATER THAN rank THEN
                PRINT rank is less than max
                SET status to '0'
                PRINT process became passive
            ELSE IF max is EQUAL TO rank THEN
                INCREMENT round
                SET local leader to rank
            END IF
            PRINT process, round, and status
        END IF
        CALL MPI_Allreduce with MPI_MAX on every process's local leader to their global leader
END WHILE

CALL MPI_Barrier with MPI_COMM_WORLD
IF rank is EQUAL TO global leader THEN
```

```
        PRINT process is leader
END IF
CALL MPI_Barrier with MPI_COMM_WORLD

CALL MPI_Finalize
RETURN 0
```

How to Run the Program

1. Navigate to the directory that contains **franklin_ring.exe** with your OS's terminal / command-line.
    a. Type **mpirun -n [INSERT NUMBER OF PROCESSES HERE] franklin_ring.exe** into the terminal / command-line and press enter.
        i. The specified number of processes must be, at least, 2.

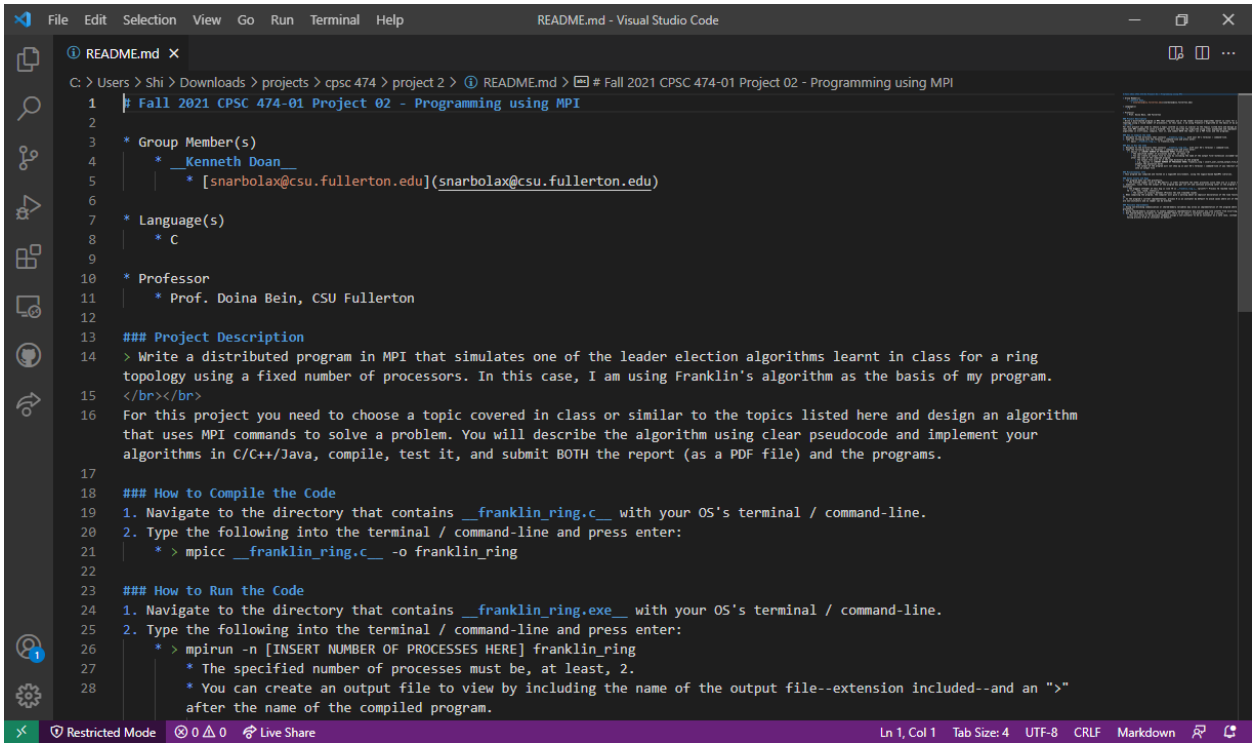
You can create an output file to view by including the name of the output file--extension included--and an ">" after the name of the compiled program.

- e.g. "**mpirun -n [INSERT NUMBER OF PROCESSES HERE] franklin_ring > insert_user_custom_output_file_here.txt**", without the quotation marks
    - the output file will be in the same directory of the program
    - the output of the program will not show up in your OS's terminal / command-line if you redirect its output into an output file

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

Snapshots

Group Member(s)



Code I/O

Included as:

- **sample_output_n2.txt**
    - Output file of **franklin_ring.exe**, where n = 2 when running the program
- **sample_output_n5.txt**
    - Output file of **franklin_ring.exe**, where n = 5 when running the program