Mason Godfrey (mgodfrey@csu.fullerton.edu)

Bijaya Shrestha (sthavjay@csu.fullerton.edu)

CPSC 474

Project 1 Submission

SUMMARY

Contents

This report contains both the pseudocode for every file (main.cpp, and the headers [header, OutputArray, AlgorithmVerify, and AlgorithmCalculate]) and snapshots. The pseudocode is perhaps more detailed than necessary, but provides a somewhat detailed explanation of how our program works. The first snapshot contains our group member names, and the other two provide example runs.

Running Code

This program was made and tested using eclipse software.

To run the program with test parameters:

(1) Place all .h and .cpp files in the same directory.

(2) Build the project.

(3) Run the program.

To run the program with custom parameters:

(1) Change the row and col variables in header.h to reflect the size of the array(s) you want to test.

(2a) To test the AlgorithmCalculate function, change ex1Events in main.cpp to have the string values you would like to use.

(2b) To test the AlgorithmVerify function, change ex1Clock, ex2Clock, and/or ex3Clock to have the int values you would like to use.

(3) Ensure that the row and col values are accurate for the new array you are testing, and comment out any array/function you are not testing. If you wish to test multiple arrays with different sizes, please do so in separate program runs.

(4) Complete all steps in the 'run the program with test parameters' section.

PSEUDOCODE FOR ALL FILES

1) Pseudocode for main.cpp

a. Call algorithm calculate using the matrix of events

b. Call algorithm verify using each of the three LC-value matrices

c. Return

2) Pseudocode for header.h

a. Create a string event matrix of events as a 2D array

b. Create three matrices with LC-values as 2D arrays

c. Specify the number of columns and rows for all 2D arrays

3) Pseudocode for OutputArray.h

a. If user passed in a message, output the message

b. Output the array the user passed in

c. Return

4) Pseudocode for AlgorithmVerify.h (example 2, 3, 4).

a) Pass in LC-values as a 2D array
b) Create a string event matrix of events as a 2D array
c) While we're checking to see if our LC-values are possible
    a. For every column
        i. For every row
            1. If we're the next number in the sequence
                a. Increment sequence
                b. We're still adding values into event matrix
            2. If our current sequence number is larger than max
                a. Set max equal to this sequence number
d) If sequence is not equal to max
    a. Output that the sequence is incorrect
    b. Return with error code -1
e) Initialize the event matrix equal to NULL
f) For every row
    a. Reset previous element number
    b. For every column
        i. If previous element number was just reset
            1. If our current index is not sequential with previous element number
                a. Mark index as a receiver
            2. Else
                a. if we're in column one but our event sequence is not 1
                    i. Mark index as a receiver

3. Set previous as the current index's LC-value

g) While we're still updating senders/receivers
   a. Keep track of the receiver number we're on
   b. Reset sender's logic clock
   c. For every row and column
      i. If we're currently the smallest receiver that we haven't yet found the sender for
         1. Update the sender's logic clock
         2. Set receiver's index
   d. If we have the index for the receiver
      i. For every row and column where we haven't found the sender yet
         1. If the current index is the sender
            a. Set the sender number, receiver number
            b. Mark that we've found the sender
   e. Unflag that we've found the sender
h) For every row and column
   a. If we shouldn't be NULL and aren't marked as a sender or receiver
      i. Give the index a letter value and increment letter
i) Output the entire event matrix
j) Return

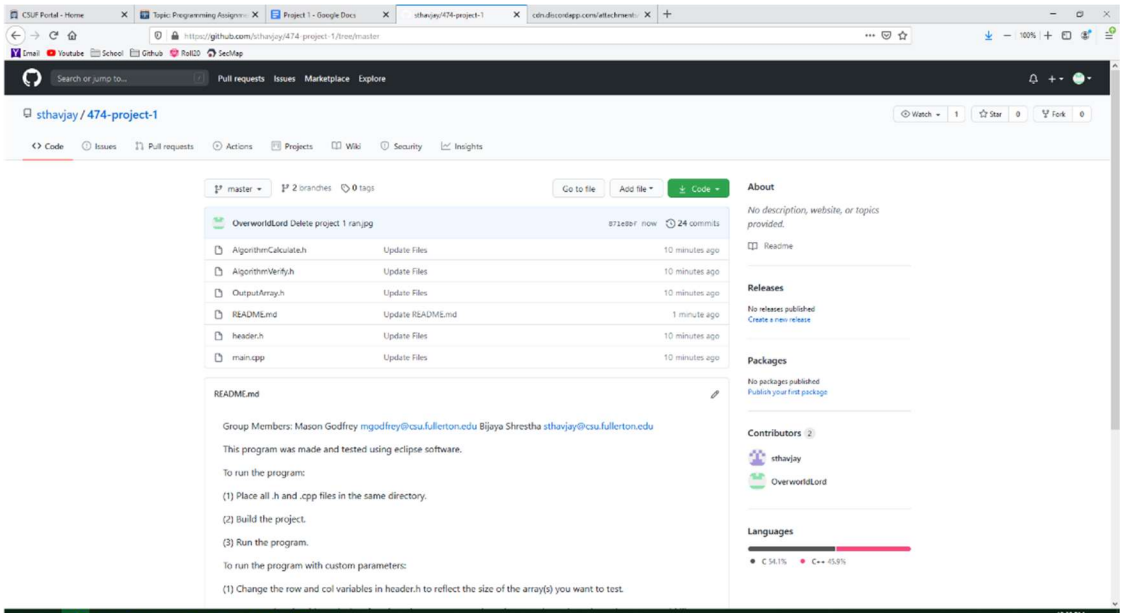
5) Pseudocode for AlgorithmCalculate.h (example 1)

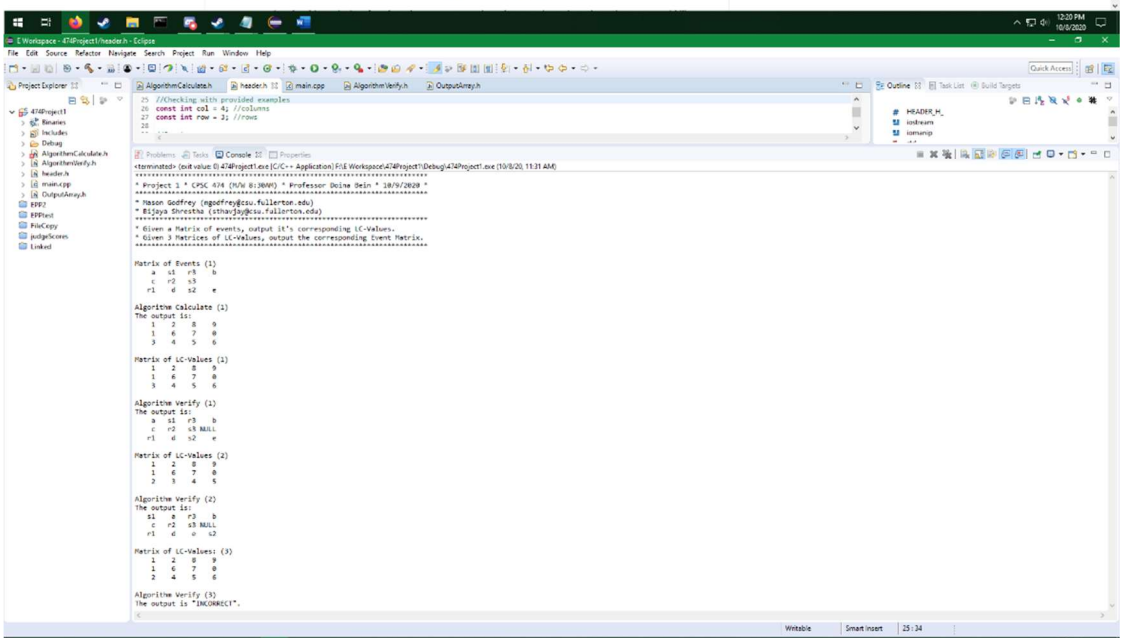
a) Pass in matrix of events
b) Create LC-value matrix

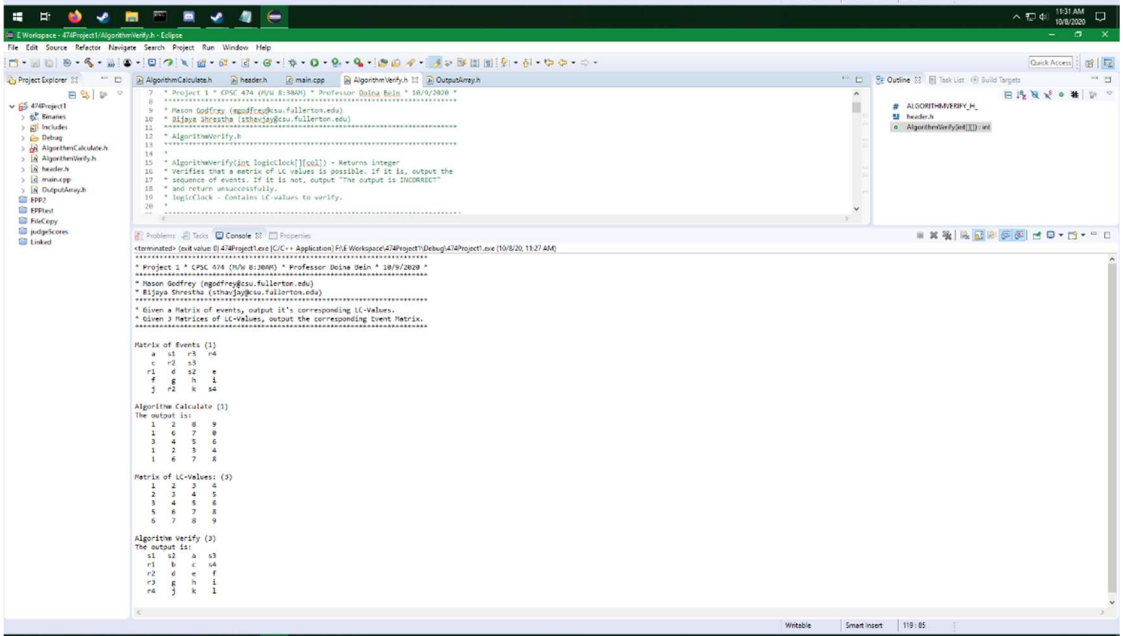
c) Initialize logicClocks to 0
d) Initialize an array that keeps track of each row's response point
e) While we're still making changes
    a. For each row
        i. For each column
            1. If index is not a response point or past a response point that was already found
                a. update logic clock based on index before
            2. Else if this is a response point
                a. Update the location of the response point in this row
            3. If index is a send point
                a. If there's a response point sender can reach
                    i. Set LC-value equal of response point to the maximum of it's sequential value or the sender's value plus one
                    ii. Reset the response point
            4. Set k equal to index's clock number
    b. Reset k when we change the row
    c. Check if we are done making changes
f) Output the matrix with LC-values
g) Return

SNAPSHOTS

The next page has three snapshots; The first one contains the names of our group members, the second contains the code execution with the base examples (N = 3), and the third contains the code execution with custom examples (N = 5). This image can also be found in the master branch of the repository.


sthavjay / 474-project-1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
master
2 branches
0 tags
Go to file
Add file
Code
OverworldLord Delete project 1 ran.jpg
871e0bf now
24 commits
AlgorithmCalculate.h Update Files 10 minutes ago
AlgorithmVerify.h Update Files 10 minutes ago
OutputArray.h Update Files 10 minutes ago
README.md Update README.md 1 minute ago
header.h Update Files 10 minutes ago
main.cpp Update Files 10 minutes ago
README.md
Group Members: Mason Godfrey mgodfrey@csu.fullerton.edu Bijaya Shrestha sthavjay@csu.fullerton.edu
This program was made and tested using eclipse software.
To run the program:
(1) Place all .h and .cpp files in the same directory.
(2) Build the project.
(3) Run the program.
To run the program with custom parameters:
(1) Change the row and col variables in header.h to reflect the size of the array(s) you want to test.
About
No description, website, or topics provided.
Readme
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
sthavjay
OverworldLord
Languages
C 54.1%
C++ 45.9%


* Project 1 * CPSC 474 (M/W 8:30AM) * Professor Doina Bein * 10/9/2020 *
* Mason Godfrey (mgodfrey@csu.fullerton.edu)
* Bijaya Shrestha (sthavjay@csu.fullerton.edu)
* Given a Matrix of events, output it's corresponding LC-Values.
* Given 3 Matrices of LC-Values, output the corresponding Event Matrix.
Matrix of Events (1)
Algorithm Calculate (1)
The output is:
Matrix of LC-Values (1)
Algorithm Verify (1)
The output is:
Matrix of LC-Values (2)
Algorithm Verify (2)
The output is:
Matrix of LC-Values: (3)
Algorithm Verify (3)
The output is "INCORRECT".


* Project 1 * CPSC 474 (M/W 8:30AM) * Professor Doina Bein * 10/9/2020 *
* Mason Godfrey (mgodfrey@csu.fullerton.edu)
* Bijaya Shrestha (sthavjay@csu.fullerton.edu)
* AlgorithmVerify.h
* AlgorithmVerify(int logicClock[][col]) - Returns integer
* Verifies that a matrix of LC values is possible. If it is, output the
* sequence of events. If it is not, output "The output is INCORRECT"
* and return unsuccessfully.
* logicClock - Contains LC-values to verify.
* Given a Matrix of events, output it's corresponding LC-Values.
* Given 3 Matrices of LC-Values, output the corresponding Event Matrix.
Matrix of Events (1)
Algorithm Calculate (1)
The output is:
Matrix of LC-Values: (3)
Algorithm Verify (3)
The output is: