CPSC 474 Project 1

Distributed Computing - Lamport’s logical clocks

Group Members:

1. Cassandra Kobayashi
2. Chanmolita Nuon

Graphical user interface, application, Word

Description automatically generated

The Verify algorithm will:

1. Go through each index and replace any:
   1. 0’s with NULL
   2. EXTERNAL RECEIVE events with rk
2. Go through each index and:
   1. find the corresponding EXTERNAL SEND events for every rk
      1. replace them with sk then assign the corresponding k-value
      2. if there are RECEIVE events with no corresponding SEND event the program ends
3. Go through each index and replace any:
   1. Remaining INTERNAL events with a letter

# **Pseudocode Verify Algorithm:**

**def** verify(vector&dataVect):

[p\*e] = dataVect.size // p = processes e = events

N= 5, M=25 // max values

vectorH(N\*M, 0)

temp

proccesses = p

alpha[24] = a to z //excludes s and r

**while** (processes > 0) **do**

k = 0

index = 0

**for** i = 0 to (p -1) **do**

**for** j = 0 to (e-1) **do**

if (data[i][j] == 0)

vectorH[i][j] = NULL

else if (data[i][j] != 1 && j == 0)

vectorH[i][j] = rk

else if (data[i][j] != data[i][j-1]+1 && data[i][j] != 0 && j != 0)

vectorH[i][j] = rk

**end if**

**end for**

**end for**

**for** i = 0 to (p -1) **do**

**for** j = 0 to (e-1) **do**

if (vectorH[i][j] == rk)

vectorH[i][j] = r+(k\_value)

temp = data[i][j]

**for** i = 0 to (p -1) **do**

**for** j = 0 to (e-1) **do**

if (data[i][j] == (temp-1))

vectorH[i][j] = s + (k\_value)

k++

break

**end if**

if (break)

exit (1)

**end if**

**end for**

**end for**

**end if**

**end for**

**end for**

**for** i = 0 to (p-1) **do**

**for** j = 0 to (e-1) **do**

if (vectorH[i][j] == 0)

vectorH[i][j] = alpha[index % 24]

index++

**end if**

**end for**

**end for**

processes --

**end while**

The calculate algorithm will:

1. Go through each index and replace any:
   1. NULL with 0
   2. First event and is an internal or sent event: LC = 1
   3. First event and is a receive event LC = k + 1, k is LC-value of the send event that correspond to that receive event
   4. Not first event and is an internal or send event LC = k + 1, k is the LC-values of the element before
   5. Not first event and is a receive event, LC = max(element before, LC of sent event corresponding to receive event) + 1

# **Pseudocode Calculate Algorithm:**

**def calculateAlgorithm(string data[process][event])**

**vector<int> sPosition;**

**int lc[process][event];**

**int run = process;**

**while(run > 0)**

**for (int i = 0; i < process; i++)**

**for(int j = 1; j < event+1; j++)**

**if( event isn’t already assign || data[i][j] != “NULL”)**

**if ( first event && event != receive)**

**{**

**lc[i][j] = 1;**

**if(event == sent)**

**add the int to sPosition**

**add lc value to sPosition**

**}**

**else if(event != receive)**

**{**

**lc[i][j] = lc[i][j-1]+1**

**if (event == sent)**

**add the int to sPosition**

**add lc value to sPosition**

**}**

**else**

**{**

**for (int k = 0; k < sPosition.size(); k+=2)**

**{**

**if (receive correspond to s )**

**{**

**if (sent lc-value > lc value of event just before receive)**

**lc[i][j] = sent lc-value + 1;**

**else**

**lc[i][j] = lc of event just before + 1**

**}**

**}**

**}**

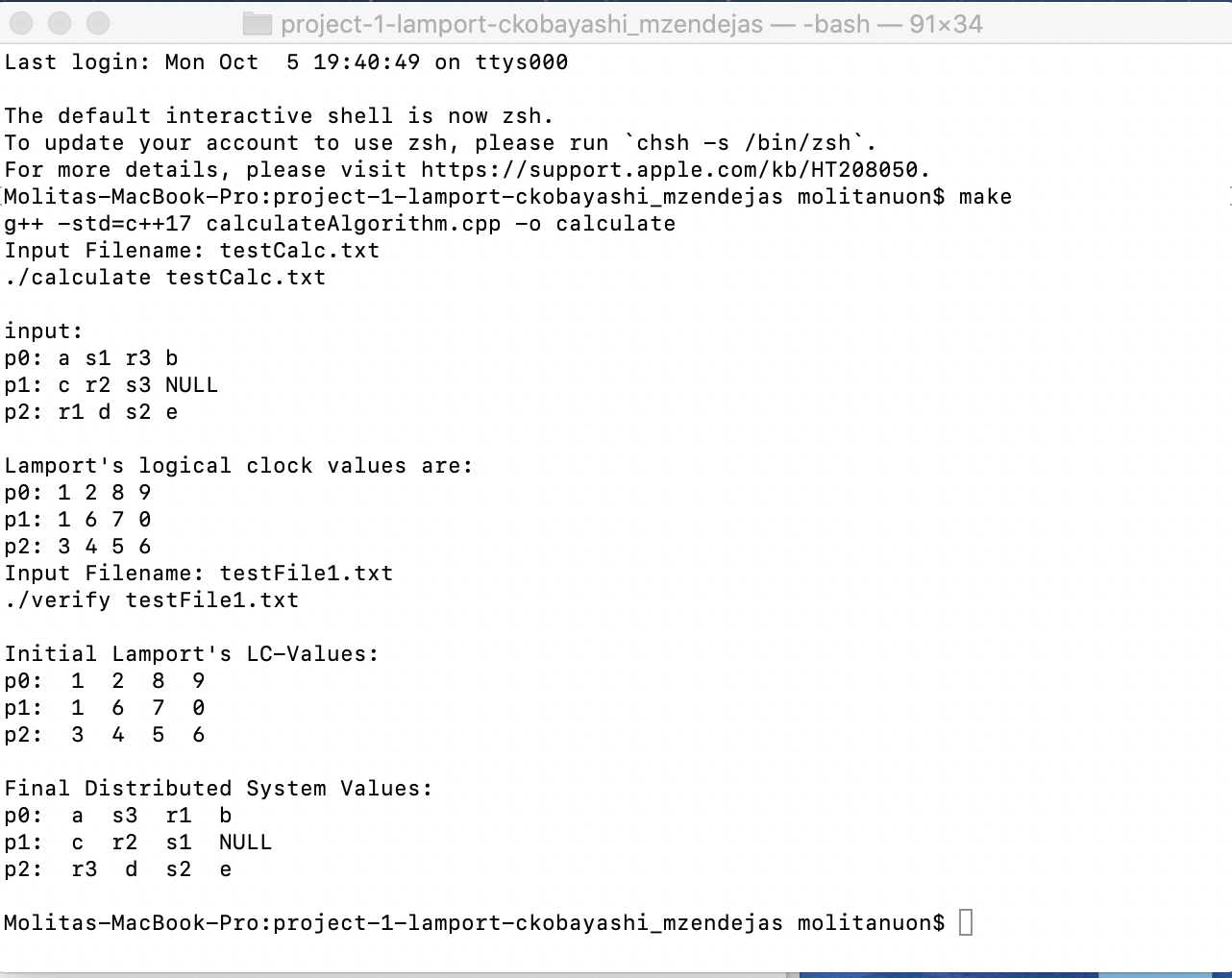
**decrement run**

# **Parameters Needed to Execute the Verify Algorithm and Calculate Algorithm:**

**The whole program run sequentially**

In order to execute the program correctly use: **MAKE** command

To compile and run the program use: **make** and then you will ask for the test file name for the **calculate algorithm**. **(**i.e. **testCalc.txt )** then you will be ask for the test file name for the **verify algorithm (ie testfile1.txt)**

****

**TO RUN THE ALGORITHM SEPARATELY**

**VERIFY ALGORITHM**

\* You can also compile using **g++ verifyAlgorithm.cpp -o verify**

\* To run use ./verify <test filename> i.e. **./verify testfile1.txt**

Text, letter

Description automatically generated

To print an output file use: **MAKE OUTPUT** command

Input the name of the test file and then name the output file i.e. **output\_testfile1.txt**

The file will be saved in the outputFiles folder

Text

Description automatically generated

\* You can also use ./verify testfile1.txt > outputFiles/<output\_filename>

i.e. **./verify testfile1.txt > outputFiles/output\_file1.txt**

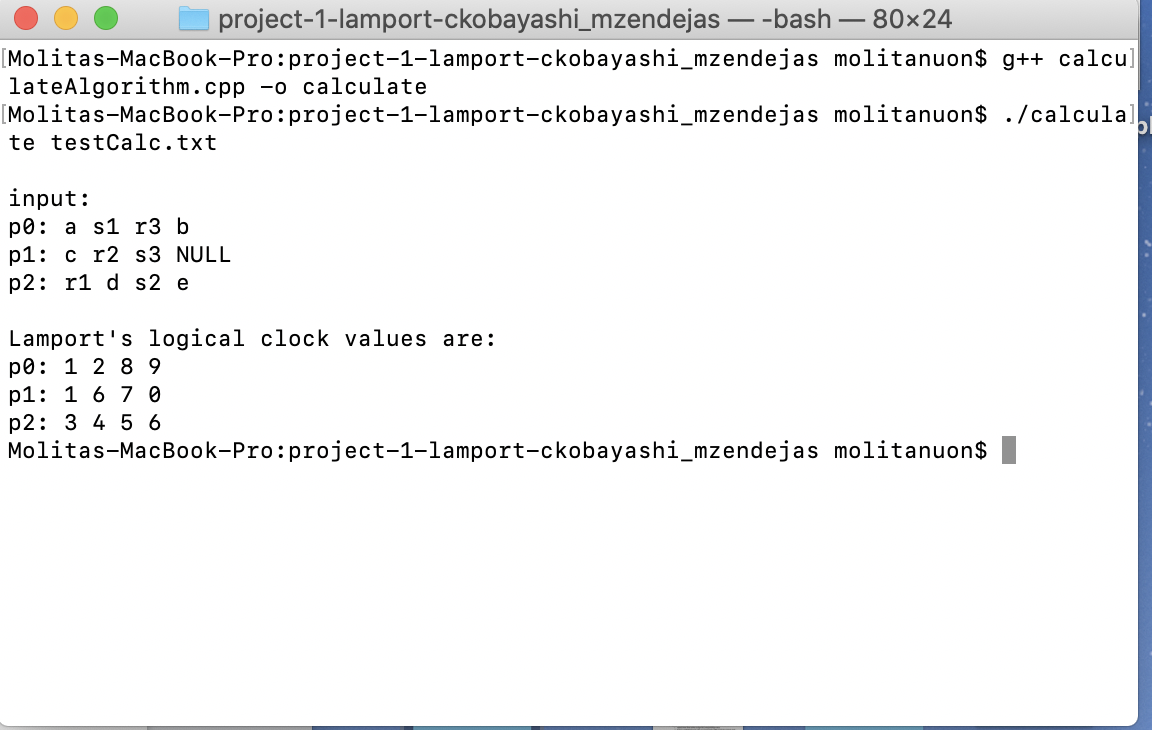
**Screenshots:**

|  |  |
| --- | --- |
| Example 1: | Example 2: |
| Example 3: | Example 4: |

|  |  |
| --- | --- |
| To run own tests:  A picture containing calendar  Description automatically generated | The program will read the input file in this order:   1. The first digit as the number of processors and the second as the number of events. 2. The matrix contains the Lamport’s Logical Clock Values. |

**CALCULATE ALGORITHM**

\* You can also compile using **g++ calculateAlgorithm cpp -o calculate**

\* To run use ./verify <test filename> i.e. **./calculate testCalc.txt**

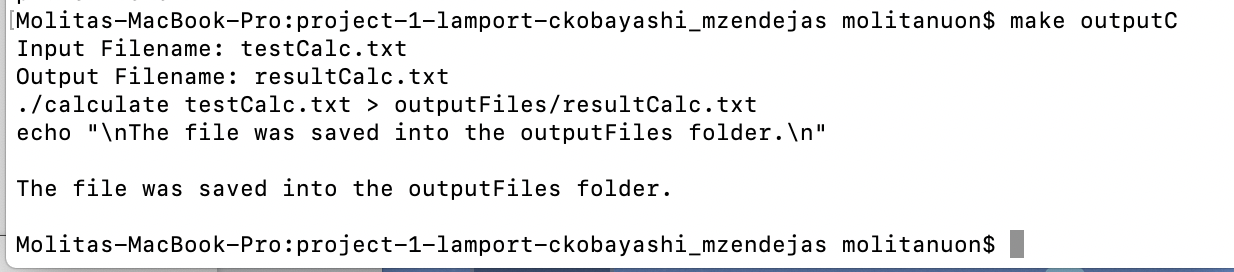
To print an output file use: **MAKE OUTPUT** command(make outputC)

Input the name of the test file and then name the output file i.e. resultCalc**.txt**

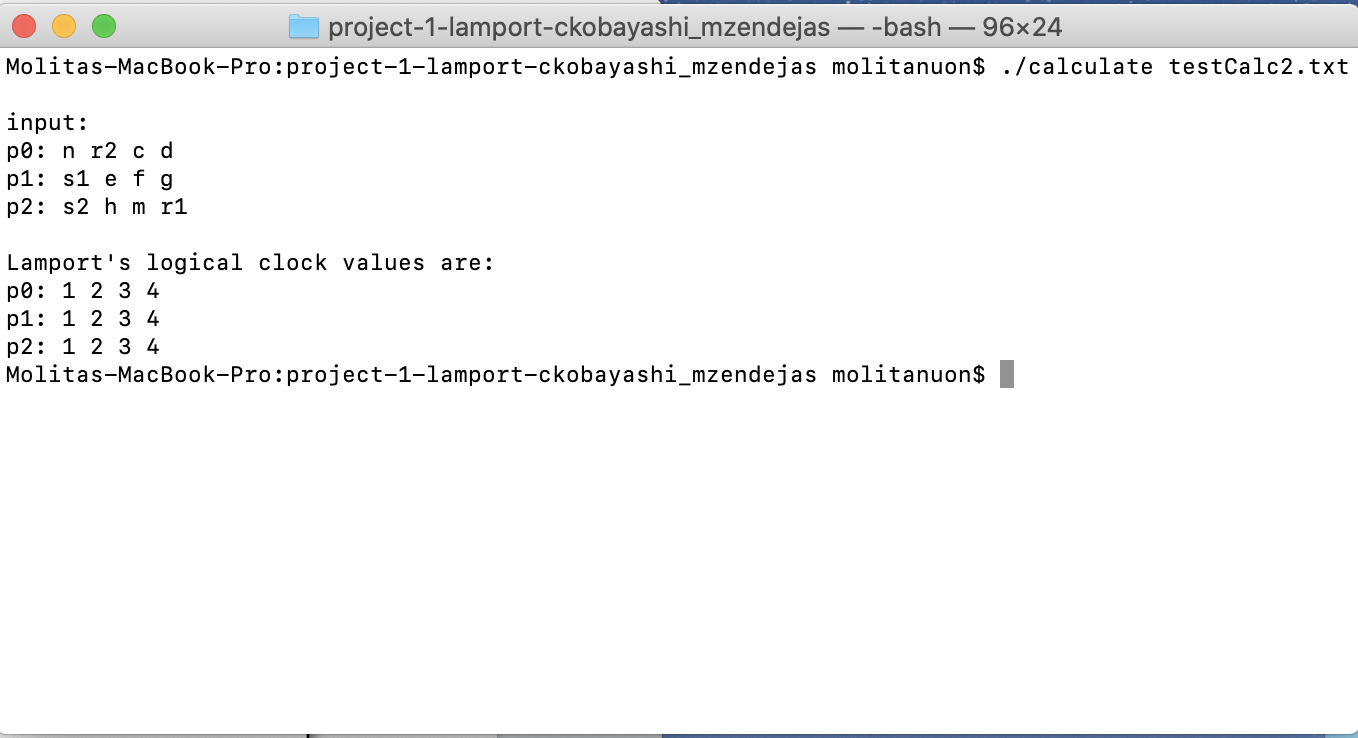
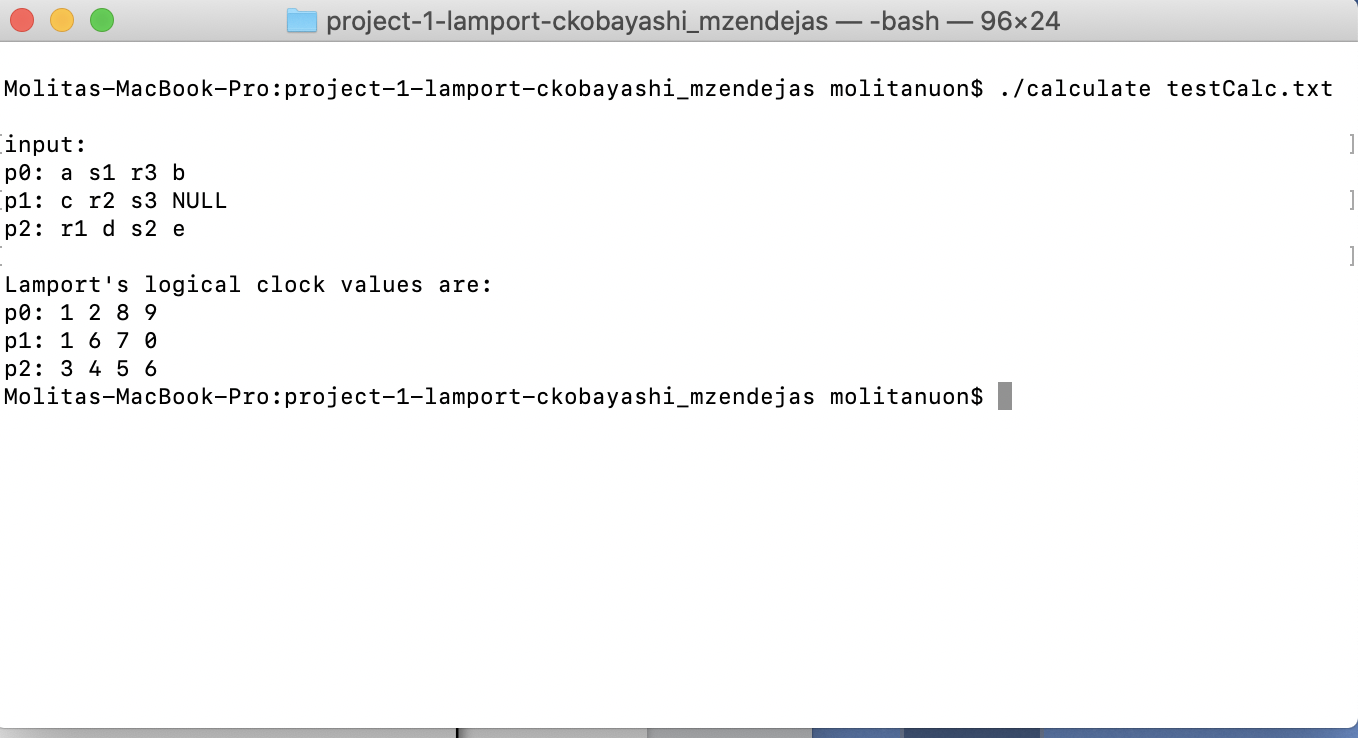
The file will be saved in the outputFiles folder

\* You can also use ./calculate testfile1.txt > outputFiles/<output\_filename>

i.e. **./calculate testCalc.txt > outputFiles/resultCalc.txt**



Screenshots

Example 1 Example 2 

To run your own test:

* to go one of the testCalc input files which can be found in the folder inputFiles
* Change the number of process and events making sure there are space between them
* You can change events and process but make sure there are spaces between them as shown
* The program will read the input files as such:
  + Skip the “Process” read the value after and skip “Events”, read the value after
  + Then read whole matrix and store it