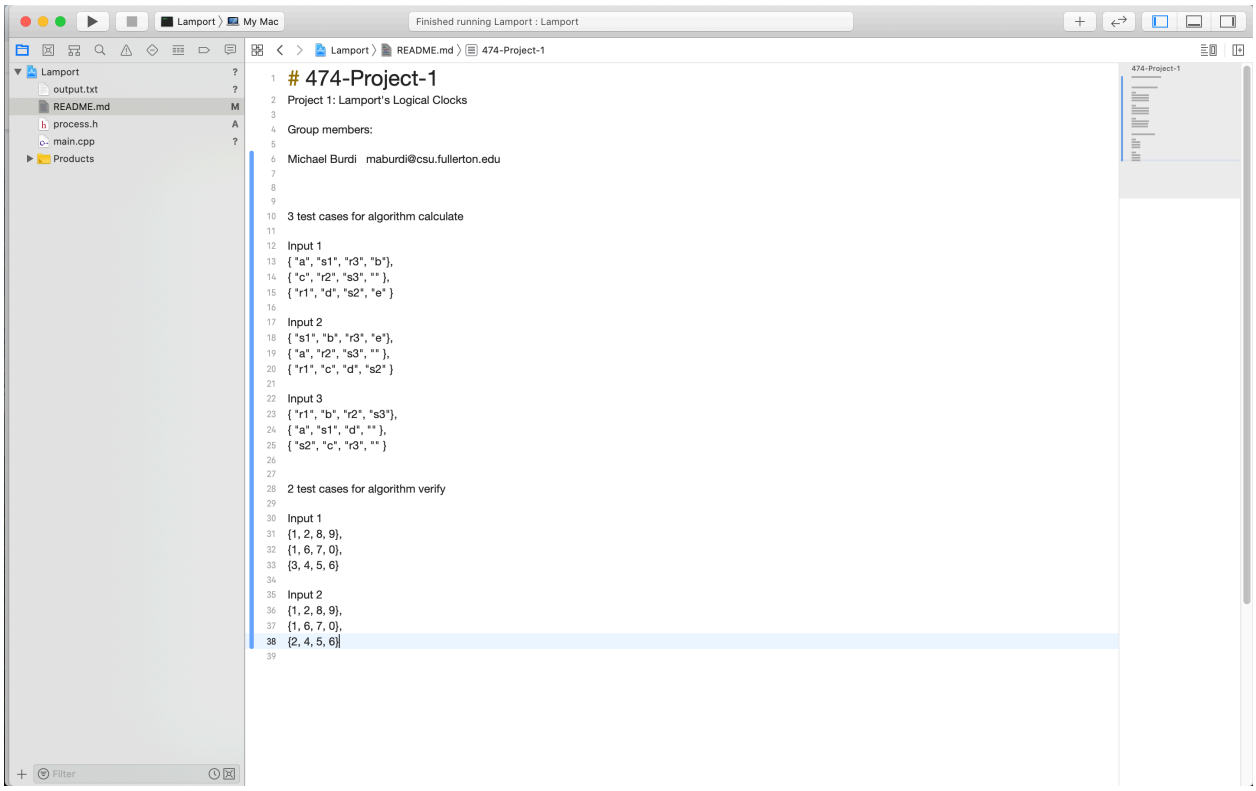
Project 1: Lamport Logical Clocks

Michael Burdi
maburdi@csu.fullerton.edu



Lamport Calculate Pseudo-code

```
def calculate (string[N][M] events)

    int[N][M] clock_values;

    int send_events[9];

    // Find all the send events and save their locations in an array
    for i = 0 to N
        for j = 0 to M

            if not the first event (j > 0)
                k = previous clock value (clock_values[i][j-1]) + 1
            else if not clock_value i,j not set yet
                k = 1
            // else keep the clock value > 1; this is a receive event

            if event.type = 's' (send)
                send_events[events[N][M][1]] = k; // Event at i,j get event id and save for later
        end
    end

    // Now we loop through all the events and look for a send event for each receive event

    for each receive event r      // we need to match every receive with a send
        for i = 0 to N
            for j = 0 to M

                if j > 0
                    k = clock_values[i][j-1] + 1
                else if not clock_values[i][j]
                    k = 1

                if event.type = 's'
                    send_events[events[N][M][1]] = k; // Event at i,j get event id and save for later
                else if event.type = 'r' and r.id = event.id
                    clock_values[i][j] = max (k, send[r.id])
            end
        end
    end

    return clock_values
```

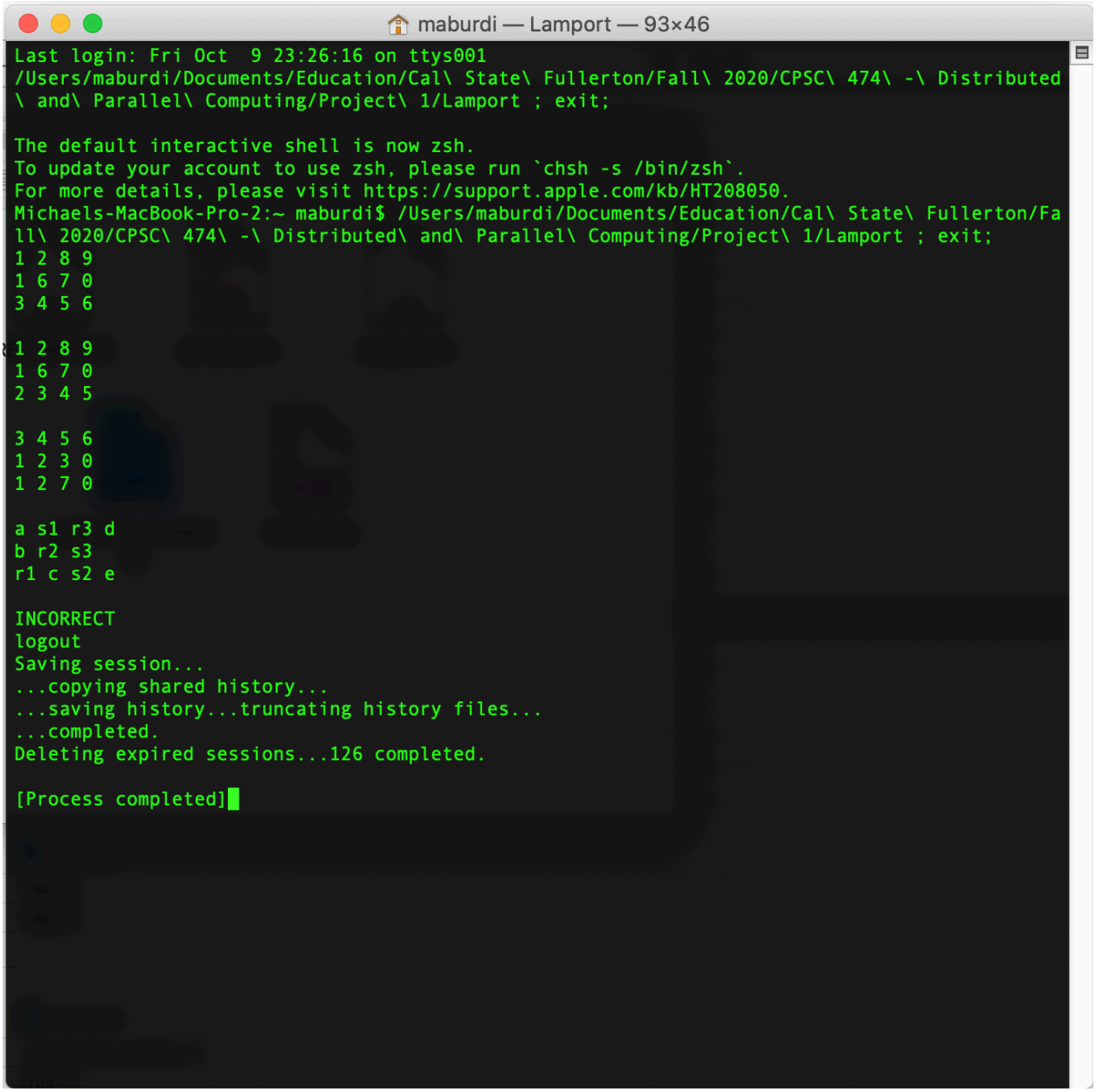
Lamport Verify Pseudo-code

```
def verify (int[N][M] clock_values)

    string[N][M] output;   // hold output matrix configuration

    // get an array of length 24 of pointers to string
    // the index of the array is the clock value and the pointer points to the output string
    string[24] values

    int max

    for i = 0 to N
        for j = 0 to M
            max = clock[i][j] > max : clock[i][j]       // keep track of the highest clock value

            if first event in process (j == 0)
                if clock == 1 // must be send or internal
                    output[i][j] = "si"

                    /* store a  pointer to this location in values array with index corresponding to the value
                    we found at clock[i][j]. Only update if 0 for "si" */

                else if clock > 1 // must be receive as first event in process lc > 1
                    output[i][j] = "si"

                    /* store a  pointer to this location in values array with index corresponding to the value
                    we found at clock[i][j]. This time we update if 0 or "si". "r" is of more interest */
            else // not first event in process
                if clock – clock previous == 1  // interval of 1. Must be send or internal
                    output[i][j] = "si"

                    /* store a  pointer to this location in values array with index corresponding to the value
                    we found at clock[i][j]. Only update if 0 for "si" */

                else if clock  - clock previous > 1 // must be receive as previous event in process leaves a gap
                    output[i][j] = "si"

                    /* store a  pointer to this location in values array with index corresponding to the value
                    we found at clock[i][j]. This time we update if 0 or "si". "r" is of more interest */
        end
    end

    int count = 1;       // give each receive event an id
    for i = 0 to max    // loop through all the clock values in order
        if values[i] == null pointer        // There was a value skipped. This output is not correct
            print("Incorrect")
            return
        else
            values[i] = "r" + the id (count)
            values[i-1] = "s" + the id    // this event right before must be the send event because it has lc – 1
            count++
    end
```

```
// for all internal events
char letter = ‘a’    // start with ‘a’ and label all internal events with a increasing letter
for i = 0 to M              // now loop over across columns
      for j = 0 to N
            if output[i][j] == “si”  // its id has not been updated so this is an internal event
                  output[i][j] = letter++ // a -> b -> c -> d -> etc.
      end
end

return output
```

How to run the code

1. Run the Unix EXE file by double-clicking the file
2. [Alternatively] Run the EXE from the Terminal using ./Lamport command while in the correct directory.


maburdi — Lamport — 93×46
Last login: Fri Oct 9 23:26:16 on ttys001
/Users/maburdi/Documents/Education/Cal\ State\ Fullerton/Fall\ 2020/CPSC\ 474\ -\ Distributed\ and\ Parallel\ Computing/Project\ 1/Lamport ; exit;
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Michaels-MacBook-Pro-2:~ maburdi$ /Users/maburdi/Documents/Education/Cal\ State\ Fullerton/Fall\ 2020/CPSC\ 474\ -\ Distributed\ and\ Parallel\ Computing/Project\ 1/Lamport ; exit;
1 2 8 9
1 6 7 0
3 4 5 6
1 2 8 9
1 6 7 0
2 3 4 5
3 4 5 6
1 2 3 0
1 2 7 0
a s1 r3 d
b r2 s3
r1 c s2 e
INCORRECT
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.
Deleting expired sessions...126 completed.
[Process completed]