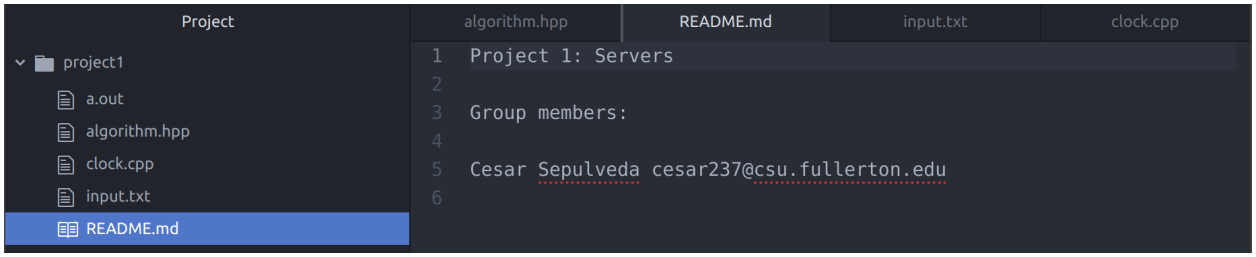
CPSC 474 Project 1

Group Members:
Cesar Sepulveda cesar237@csu.fullerton.edu

Screenshot of README file:



Description/How to Run the Program:

The program is written in C++ and uses three different files to run: main.cpp, algorithms.hpp, and input.txt. The file main.cpp is used to run the program's menu is is the file that should be compiled. The text file input.txt is where the program grabs the matrix to test the algorithms. The input should be written in the same format as what is already in input.txt. Lastly, algorithm.hpp houses all of the helper functions as well as the two algorithms calculate and verify. When running the program the first thing it will do is bring up the menu: 1 Calculate, 2 Verify, 0 Exit. You choose the option by entering the corresponding number. After choosing which algorithm to run it will ask you to enter the number of rows (n), followed by the number of columns (m) for the matrix. After entering this information, the algorithms will run and the program will output the original matrix and the updated one. In the case of the verify algorithm if the original matrix is incorrect the program will simply output "incorrect".

Pseudocode calculate:

```
void calculate():
  integer n, m                          //Initializing variables.
  integer s = 0
  integer r = 0
  boolean finished = false
  string array senders {"s1", "s2", …, "s9"}        //Initializing arrays to keep
  string array receivers {"r1", "r2", …, "r9"}      //track of  sends and receives.
  Output: "How many rows (N)?"
  input n                          //Asking user to enter the amount of rows(n) and columns(m).
  Output: "How many columns (M)?"
  input m
```

```
integer array count[n]
for i = 0, i < n, i = i+1:
  Count[i] = 0                    //Filling the array with 0.
end for
vector of string vectors vec = readInput(n,m)        //Calling the readInput function.
for i = 0, i < size of num, i = i+1:              //For loops to iterate through the matrix
  matrix.push_back(vector of intagers)      //and fill every entry with 0.
  for j = 0, j < size of num[i], j = j+1:
    matrix[i].push_back(0)
  end for
end for
while finished is equal to false:
  for i = 0; i < vec.size(); i = i+1:    //For loops to iterate through the matrix.
    for j = 0; j <  vec[i].size(); i = i+1:        //If statements to find the LC values of events.
      if vec[i][j].length() is equal to 1 and count[i] is equal to j:
        if j is equal to 0:       //If statement in case event is the first item in the row.
          matrix[i][j] = 1      //Adding the event LC value to the matrix.
          count[i] = count[i]+1
        end if
        else:
          matrix[i][j] = matrix[i][j-1]+1      //Adding the event LC value to the matrix.
          count[i] = count[i]+1
        end else
      end if              //If statements to find the LC value for send values.
      if vec[i][j].find("s") is equal to 0 and count[i] is equal to j:
        if j is equal to 0:       //If statement in case the send value is the first item in the row.
          matrix[i][j] = 1;               //Adding the send LC value to the matrix.
          count[i] = count[i]+1
        end if
        else:
          matrix[i][j] = matrix[i][j-1]+1      //Adding the send LC value to the matrix.
          count[i] += 1
        end else
      end if
      if vec[i][j].find("r") is equal to 0 and count[i] is equal to j:
        if vec[i][j].compare(receivers[r]) is equal to 0:
          for k = 0; k < vec.size(); k = k+1: //For loops to iterate through the matrix.
            for l = 0; l <  vec[k].size(); l = l+1:
              if vec[k][l].compare(senders[s]) is equal to 0:
                if j is equal to 0://If statement in case the receive value is the first in the row.
```

```
                matrix[i][j] = findMax(0,matrix[i][j-1])+1
                count[i] = count[i]+1
              end if
              else:          //Adding the receive LC value to the matrix.
                matrix[i][j] = findMax(matrix[i][j-1],matrix[k][l])+1
                count[i] = count[i]+1
              end else
            end if
          end for
        end for
        s = s+1              //Iterating senders and receivers
        r = r+1
      end if
    end if          //If statement to handle NULL values.
    if vec[i][j].length() is equal to 4 and count[i] is equal to j:
      count[i] = count[i]+1
    end if
  end for
end for
finished = true    //Checking the conditon to exit the while loop
for int i = 0; i<n; i = i+1:
  if count[i] is not equal to vec[i].size():
    finished = true
  end if
end for
end while
display the vec              //Displaying the original matrix.
display matrix               //Displaying the updated matrix.
exit program
```

Pseudocode verify:

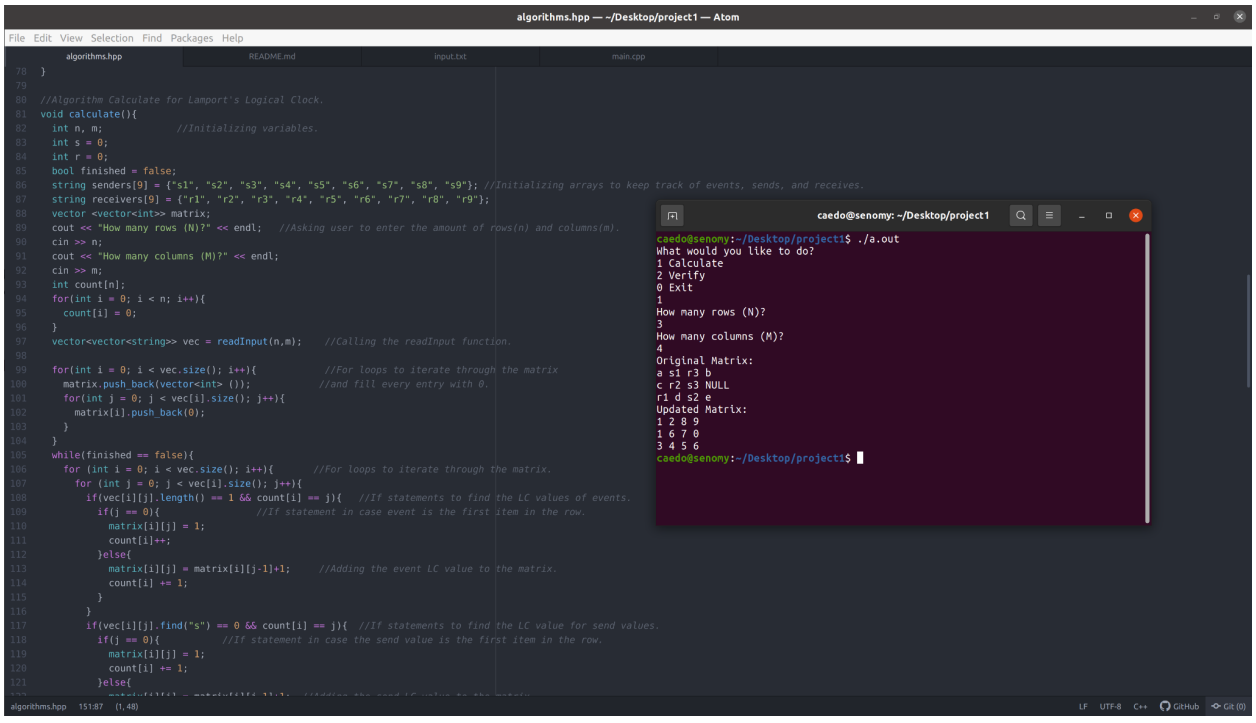
```
void verify():
  integer n, m                        //Initializing variables.
  vector of string vectors matrix
  integer s = 0
  integer r = 0
  integer e = 0
  boolean found
  string array senders {“s1”, “s2”, …, “s9”}        //Initializing arrays to keep
  string array receivers {“r1”, “r2”, …, “r9”}      //track of events, sends, and receives.
```

```
string array events {“a”, “b”, …, “z”}
Output: “How many rows (N)?”
input n                          //Asking user to enter the amount of rows(n) and columns(m).
Output: “How many columns (M)?”
input m
vector of string vectors vec = readInput(n,m)        //Calling the readInput function.
vector of integer vectors num = changeToInt(vec) //Calling the changeToInt function
for i = 0, i < size of num, i = i+1:              //For loops to iterate through the matrix
  matrix.push_back(vector of strings)       //and fill every entry with NULL.
  for j = 0, j < size of num[i], j = j+1:
    matrix[i].push_back(“NULL”)
  end for
end for
for i = 0, i < size of num, i = i+1:    //For loops to find the recieve values in the matrix.
  for j = 0, j < size of num[i], j = j+1:
    if j > 0 and num[i][j-1]+1 is not equal to num[i][j] and num[i][j] is not equal to 0:
      matrix[i][j] = receivers[r]       //Adding a recive value to to the matrix
      r = r+1                           //Incrementing r.
    end if
    else if j is equal to o and num[i][j] is not equal to 1:
      matrix[i][j] = receivers[r]       //Adding a recive value to to the matrix
      r = r+1                           //Incrementing r.
    end else if
  end for
end for
 for i = 0, i < size of matrix, i = i+1://For to locate where the send values are in the matrix.
 found = false;

  found = false
  for j = 0, j < size of matrix[i], j = j+1:
    if matrix[i][j].find(“r”) == 0:
      for k = 0, k < size of num, k = k+1:   //Finding the location of previous receive values.
        for l = 0, l < size of num[k], l = l+1:
          if num[i][j] - num[k][l] == 1:       //Finding the correct time slot
            matrix[k][l] = senders[s];         //Adding the send value to the matrix.
            s = s+1
            found = true
          end if
        end for
      end for
```

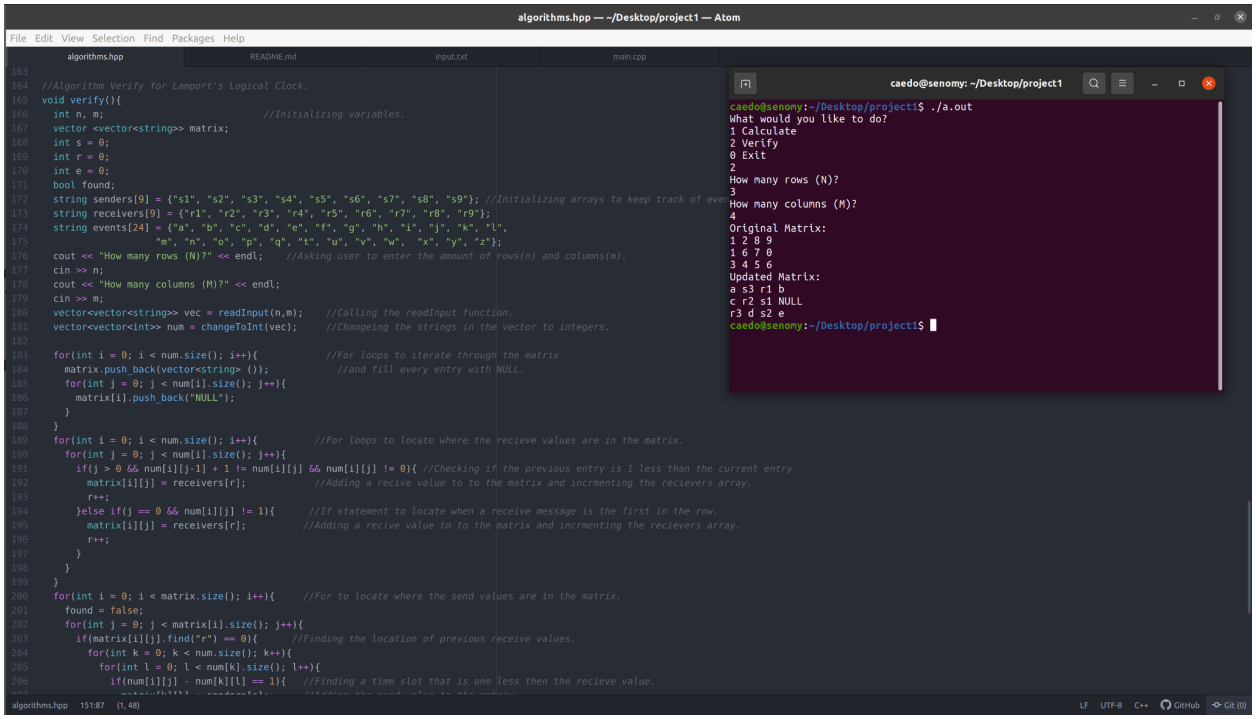
end if
end for
if found is equal to false: //**Outputs incorrect if the matrix is not solvable**
Output: “incorrect”
exit program
end if
end for
for i = 0, i < size of matrix, i = i+1: //**For loops to fill ot the rest of the matrix with events.**
for j = 0, j < size of matrix[i], j = j+1:
If matrix[i][j].find(“r”) is not equal to 0 and matrix[i][j].find(“s ”) is not equal to 0 and
num[i][j] is not equal to 0: //**Checking that the value is not a send, receive, or null.**
matrix[i][j] = events[e] //**Adding the event to the matrix**
e = e+1
end if
end for
end for
display the vec //**Displaying the original matrix.**
display matrix //**Displaying the updated matrix.**
exit program

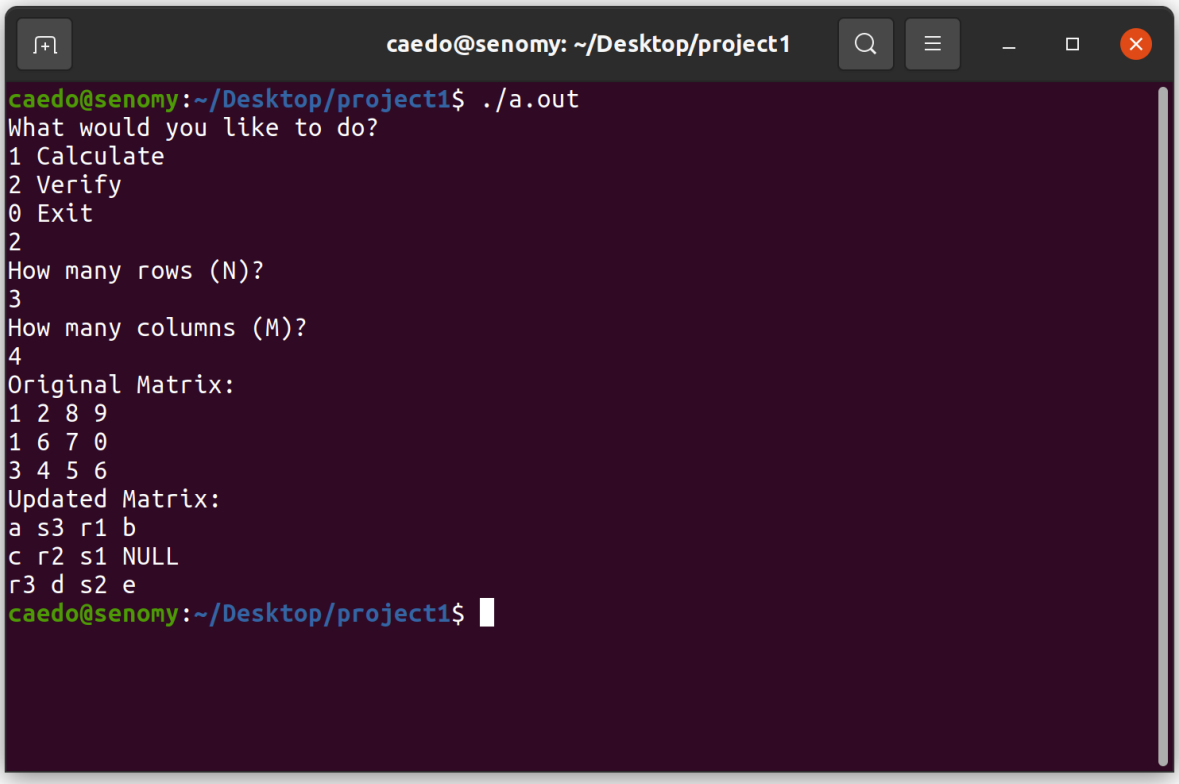
Screenshots of the calculate algorithm:



```
caedo@senomy:~/Desktop/project1$ ./a.out
What would you like to do?
1 Calculate
2 Verify
0 Exit
1
How many rows (N)?
3
How many columns (M)?
4
Original Matrix:
a s1 r3 b
c r2 s3 NULL
r1 d s2 e
Updated Matrix:
1 2 8 9
1 6 7 0
3 4 5 6
caedo@senomy:~/Desktop/project1$
```

Screenshots of the verify algorithm:




caedo@senomy: ~/Desktop/project1
caedo@senomy:~/Desktop/project1$ ./a.out
What would you like to do?
1 Calculate
2 Verify
0 Exit
2
How many rows (N)?
3
How many columns (M)?
4
Original Matrix:
1 2 8 9
1 6 7 0
3 4 5 6
Updated Matrix:
a s3 r1 b
c r2 s1 NULL
r3 d s2 e
caedo@senomy:~/Desktop/project1$