CPSC-474

LAMPORT LOGICAL CLOCK & PSEUDOCODE

ALGORITHM FOR CALCULATION AND VERIFICATION

GROUP MEMBER NAMES:

1. Balwinder S. Hayer
2. Saytu Singh

Calculation: (Completed solely by Balwinder Hayer)

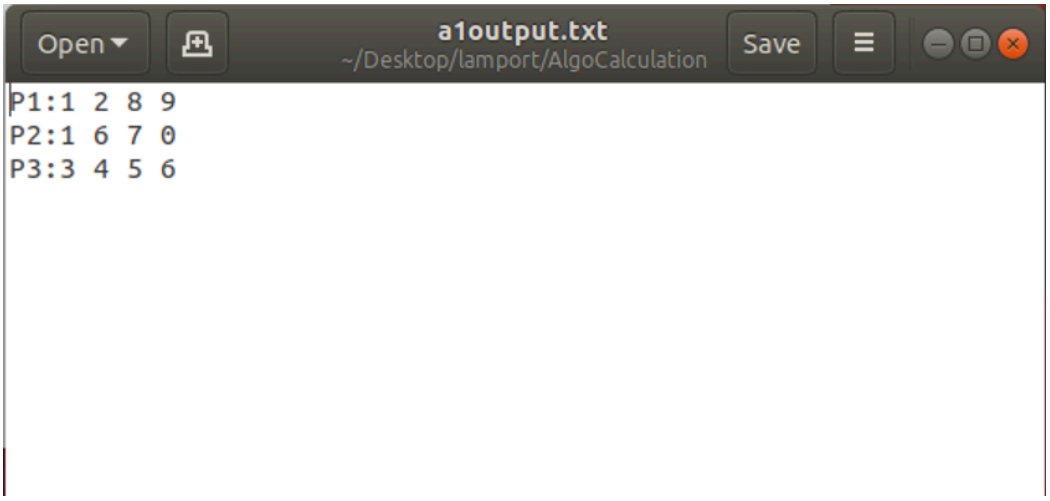
Execute commands (HOW TO RUN IT) -

```
g++ -o AlgoCalc AlgoCalc.cpp
```

```
./AlgoCalc
```

SCREENSHOTS OF PART1:

OUTPUT TEXT FILE:



Verification: (Completed solely by Saytu Singh)

Execute commands (HOW TO RUN IT):

python verify.py

NOTE: To test Example 1, 2 and 3, open the verify.py file and edit inputfile name to input_1.txt or input_2.txt or input_3.txt and then run with same commands.

SCREENSHOTS OF PART1:

EXAMPLE 1:

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$ python verify.py
a       s1      r3      b

c       r2      s3      NULL

r1      d       s2      e

parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$
```

EXAMPLE 2:

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$ python verify.py
s1      a       r3      b

c       r2      s3      NULL

r1      d       e       s2

parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$
```

EXAMPLE 3:

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$ python verify.py
INVALID
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$
```

EXAMPLES ALL TOGETHER:

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$ ls
input_1.txt  input_2.txt  input_3.txt  output_1.txt  output_2.txt  output_3.txt  verify.py
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$ python verify.py
a       s1      r3      b

c       r2      s3      NULL

r1      d       s2      e

parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$ python verify.py
s1      a       r3      b

c       r2      s3      NULL

r1      d       e       s2

parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$ python verify.py
INVALID
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/Project/verify$
```

PSEUDOCODE AT BOTTOM

PSEUDOCODE

verify.py

```
➔ #function definition of print array is defined
   for loop used for rows in the array
      nested for loop used for columns inside the rows of the
   array
      if(columns inside the rows)
            print(columns)
            new line
   #getting filename to read input values from text file
   Open(filename) as f:
      if true
            print(inputTrue)
   #initialize the final values of N & M
      N & M len is inputTrue

   #initialize the final result using finalResult Array and
   dictionaries
      finalResult = [['-1' for x in range(M)] for y in range(N)]
      send,receiveEvent,int_events = {},{},{}

#Verification of Initialization of Lamport Logical Clock
#drawing all alphabets and populating dictionaries alphabets
     for i in range
     for j in range
     if (receive true – old value > 1)
     print(i,j)
for l in range
for m in range
     #verify every column for by LogicalClock(s) -> LogicalClock® - 1
if inputTrue
     print True
elif inputTrue [l][m] == inputTrue[i][j] -1;
     print True
else
     print"INVALID"
#exit
#Populate finalResult array send receive events
from itertools import cycle
     for i in range(N):
     for j in range(M):
     if finalresult[i][j] == 1:
     printArray(finalresult)
#END
```

AlgoCalc.cpp

➔
```
struct events
// holds what type of the event it is for example: internal or
external receive
string whatTypeEvent;
int calcClockValue = -1; //holds the calculated clock value
};

void calcLampClock
//function used to calculate when completing lamport clock for
every send and internal events
int calcSendEventClock
bool finalTestAll
 int main ()
{
   n = 0;
   m = 0;
//entering input for all the processes
for
{
   int n, m;
   n++;
   m++;
//input tempClockValue
}

//for loop to hold computational calc performed by the
algorithm
{
for (int i =0; i<n; i++)
//it checks if the event is internal or send event
if(j==0).  //verifies if the first occurred event arrived at
processor

tempClockValue[i][j].calcClockValue = 1. //assigns the logical
lamport clock equals to one or = 1 for every first occurred
event
{
tempClockValue = -1; //lamport clock assigns to minus one = -1
//elseif
}

//it checks whether it belongs to receive state
 int eventSend = calcSendEventClock //calculates the event
send of logical lamport value

int k = tempClockValue[i][j-1].caclClockValue; //temp event
which occurred before the receive event in the process
else
{
```

```
      //we set value to be (-1) but can be changed when send_events
      are processed
      tempClockValue[i][j].calcClockValue = -1;
      LogicalLamportClock[i][j] = -1;
      }

      else if
      {
      //NULL meaning no value
      tempClockValue== “0”
      LogicalLamportClocl[i][j]== “0”;
      }

      Else
      {
      LogicalLamportClock[i][j]=-1;
      tempClockValue[i][j].calcClockValue=-99;
      }

      While{
      //prints out the Lamport Logical Clock Values
      while(!finalTestAll(tempClockValue, n, m,
LogicalLamportClock))
         cout<<endl<<"Lamport Logical Clock Values(LLC): "<<endl;

      //then generation of the output text file as a1output.txt
which can be saved and writes to the output file.
      ofstream outfile("a1output.txt");
      return 0;


      //send event
      //check for internal event
      if (tempClockValue[i][j].whatTypeEvent.length() == 1 ||
      tempClockValue[i][j].whatTypeEvent[0]=='s')
      //calculates the lamport logical clock value for an internal
      event
      tempClockValue[i][j].calcClockValue = tempClockValue[i][j -
1].calcClockValue + 1;

      else  //checks for receive event
      //it calculates the logical clock for receive event
      if(eventSend!=-1)

      //calculating send event clock
         int calcSendEventClock(events tempClockValue[10][10], int
   n, int m, string a)
         else
         return -1; // return it in case send event is not occurred


         else {
```

```
                        return
tempClockValue[i][j].calcClockValue;
      }


      //checking final values
      bool finalTestAll(events tempClockValue[10][10], int n, int
m, int LogicalLamportClock[10][10])
      {
            if (tempClockValue[i][j].calcClockValue == -1)

      return false;

      else
            return true;
      }
```