Traffic Light Control System

Sushil Bohara
(sb7702@nyu.edu)

New York University Abu Dhabi

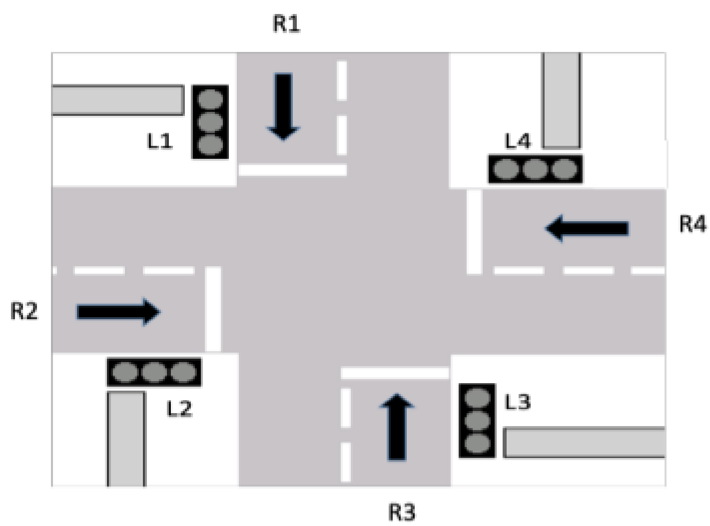
Step 1: Problem Identification and Statement

The objective is to develop a program that can control a system of traffic lights at an intersection. This program should control the specified number of traffic lights at the intersection which is declared in the program. It also must read the traffic flow information from the file every 24 hours and update the green timing of each light. At last, It should print the state of each light on the output screen.

Step 2: Gathering of Information and Input/Output Description

Relevant Information:

An intersection consists of two streets that cross at right angles. For simplicity, each street has a single lane in each direction (no lanes designated for left turn). The intersection with two roads intersected with each other is shown in the following figure.



The system has the following components:

1. Traffic semaphores (signal lights): these are standard semaphores with three lights: red, yellow, and green.
2. Traffic sensors that are embedded in each lane near the intersection to record the traffic flow for all roads (4 sensors generating 4 traffic rate values when four traffic lights are used). The sensors save the traffic rate information into a file (average number of vehicles per hour passing through a particular road in one direction).

3. The signals operate in a conventional fashion. Traffic is allowed to move on one road, say R1, and then the next (R2), alternatively across the four roads of the intersection. Assume that the four traffic lights are represented as L1, L2, L3, and L4. The system operates as follows.

a. Traffic light (L1) is green for a duration calculated based on the traffic flow rate in road R1, the other traffic lights (L2, L3, and L4) are red.
b. L1 becomes yellow for X seconds (X being a constant value). The Department of Transportation's traffic manual recommends that yellow lights are between 3 and 6 seconds long. Other traffic lights (L2, L3, and L4) remain in red state.
c. Then, traffic light L2 becomes green for a duration calculated based on the traffic flow rate in road R2. Meanwhile, L1, L3, and L4 are red.
d. Traffic light L2 becomes yellow for X seconds (X being a constant value). Other Traffic lights (L1, L3, and L4) remain in red state.
e. Then, traffic light L3 becomes green for a duration calculated based on the traffic flow rate in road R3. Meanwhile, traffic lights L1, L2, and L4 are red.
f. Traffic light L3 becomes yellow for X seconds (X being a constant value). Other traffic lights (L1, L2, and L4) remain in red state.
g. Then, traffic light L4 becomes green for a duration calculated based on the traffic flow rate in road R4. Meanwhile, traffic lights L1, L2, and L3 are red.
h. Traffic light L4 becomes yellow for X seconds (X being a constant value). Other traffic lights (L1, L2, and L3) remain in red state.
i. The next cycle starts with traffic light L1 becoming green again, and so on.

4. The green timings for the traffic lights are updated regularly based on traffic flow. Every specific duration (say 24 hours), the data(cycle length and traffic flow rates) is updated from the file. The green timings are updated based on the latest traffic condition, and the control proceeds with the updated green timings.

The green timing for each traffic light is proportional to the traffic flow rate reported for the same road, according to the following equation:

$$d_i = \frac{Q_i}{Q_T} \times C$$

Where 'd_i' is the green time for the ith traffic light, 'Q_i' represents the traffic flow (number of vehicles per hour) crossing the ith traffic light, 'Q_T' represents the total traffic flow passing through the intersection, 'C' and represents the cycle length in seconds. Cycle length is composed of the total signal time to serve all of the signal phases including the green time plus any change interval. Longer cycles will accommodate more vehicles per hour but that will also produce higher average delays.

The intersection can have more than two roads crossing each other.

Input/output Description :

The following explains how the program executes.
When the program is run, it automatically displays the colors in traffic light for infinite time.
(It runs as follows.)
Total No. of Traffic lights = 6
The light to be removed is found. Light L1 is removed.
The light to be removed is found. Light L6 is removed.
Total lights after dropping = 4

L2	Off
L4	Off

L3	Green
L5	Red

L3	Yellow
L5	Red

L5	Green
L3	Red

L5	Yellow
L3	Red

L3 Green
L5 Red
……………..
……………..
……………..

Step 3: Design of the algorithm and hand-solved problems.

Test cases :

Test case 1: ‘AddLight’ Function
When this function is called, it should add a traffic light to the program if the maximum number of traffic lights is not already achieved.

Test case 2: ‘dropLight’ Function
When this function is called, it should drop a traffic light from the program if the light to be dropped is found.

Test case 3: ‘readTrafficData’ Function
This function should be called every 24 hours to read traffic light data (cycle length and the traffic flow rates) from the file where the data is stored by a sensor.

Test case 4: ‘updateTiming’ Function
This function should also be called every 24 hours to calculate and update the green timings of all traffic lights.
For example:
If the traffic flow rate of a particular road is 100 vehicles per hour, Total traffic flow rate is 500 vehicles per hour, and Cycle length is 400 seconds,
Then Green timing = Traffic flow rate* cycle length / Total traffic flow rate
= 100*400/500 = 80 seconds
This green timing is then updated for that particular Light.

Test case 5: ‘wait’ Function:
This function should wait the program for a specified duration. For example, when the ‘green light’ is ‘on’ in any of the traffic lights, it should make the program wait for the duration equal to the green timing of that light. Similarly, if the ‘yellow light’ is ‘on’ in any of the traffic lights, it should make the program wait for the duration equal to the yellow timing of that light.

Test case 6: ‘printLightInfo’ Function:
This function should print the state of all traffic lights at the intersection.

Test case 7: getters and setters:

Getters should be able to get the values of the private members and setters should be able to modify their values.
Following default values of the members of the class TrafficLight should be obtained with the help of getters.
For L1, ID = 1, State = 1; Green Timing = 0
For L2, ID = 2, State = 1; Green Timing = 0
For L3, ID = 3, State = 1; Green Timing = 0
For L4, ID = 4, State = 1; Green Timing = 0
…………………………………………….
…………………………………………….
Following default values of the members of the class Intersection should be obtained with the help of getters.
(for 4 roads)
Default Cycle length = 400
Total Number of Traffic Lights = 4
Traffic Flow Rate of Road R1= 100
Traffic Flow Rate of Road R2= 100
Traffic Flow Rate of Road R3= 100
Traffic Flow Rate of Road R4= 100

Test case 8: run() function

This function should run the simulation continuously for infinite time.

Algorithm Design:

main () function

Create L1, L2, L3, L4, L5 and L6 as objects of class "TrafficLight"
Set the states of traffic Lights L2 and L4 to 0
Create I as an object of class "Intersection"
Add Traffic Light L1 to I
Add Traffic Light L2 to I
Add Traffic Light L3 to I
Add Traffic Light L4 to I
Add Traffic Light L5 to I
Add Traffic Light L6 to I
Print "Total No. of Traffic Lights = ", getNoOfTrafficlights of I, newline
Drop Traffic Light L1

Drop Traffic Light L6
Print "Total Lights after dropping = ", getNoOfTrafficlights of I, newline
Call the Function 'run()' of I
Return 0

HeaderFile: TrafficLight

Define class TrafficLight
Private members
ID as integer
state as integer
GreenTiming as double
NoOfTrafficlights as static integer
Public members
Default Constructor
TrafficLight()
Assign 1 to state
Assign 5 to GreenTiming
Increment NoOfTrafficlights by 1
Assign NoOfTrafficlights to ID

Non Default Constructor
TrafficLight(newstate, newGreenTiming)
Assign newstate to state
Assign newGreenTiming to GreenTiming
getID()
return ID
getstate()
return state
getGreenTiming()
return GreenTiming
getNoOfTrafficlights()
return NoOfTrafficlights
setstate(newstate)
Assign newstate to state

setGreenTiming(newGreenTiming)
Assign newGreentiming to GreenTiming
wait(seconds)
Declare waitduration as double
Find the system time and assign it to starttime
Repeat
Find the system time and assign it to endtime
Assign (endtime-starttime), in nanoseconds, to waitduration
*While (waitduration < seconds*10^9)*

printLightInfo(state, ID) {
if state is equal to 0
Print "L", ID, block of 15 characters, "Off", newline
if state is equal to 1
Print "L", ID, block of 15 characters, "Red", newline
if state is equal to 2
Print "L", ID, block of 15 characters, "Yellow", newline
if state is equal to 3
Print "L", ID, block of 15 characters, "Green", newline

(Destructor)
Initialize static variable 'NoOfTrafficlights' to 0

HeaderFile: Intersection

Declare 4 as a constant called YellowTime
Declare 8 as a constant called Max_Objects
Declare 24 as a constant called updateinterval

Define class Intersection
Private members
Cyclelength as double
NoOfTrafficlights as integer
TrafficLight_Objects as an array of objects of class 'TrafficLight' of size Max_Objects
TrafficFlowRate as an array of integers of size Max_Objects
Public members
Default Constructor
Intersection()
Assign 0 to NoOfTrafficlights

```
        Assign 400 to cycletength
        Set i to 0
              Repeat while i is less than Max_Objects
              Assign 100 to TrafficFlowRate(i)
               Increment i by 1
getNoOfTrafficlights()
    return getNoOfTrafficlights
getcyclelength()
    return cyclelength
get(address)TrafficFlowRate()
    return TrafficFlowRate(address)
setcyclelength (newcyclelength)
    Assign newcyclelength to cyclelength
setTrafficFlowRate(Array newrate)
    Set i to 0
    Repeat while i is less than Max_Objects
           Assign newrate(i) to TrafficFlowRate(i)
           Increment i by 1

 AddLight(alight)
      If NoOfTrafficlights is less than Max_Objects
             Assign alight to TrafficLight_Objects(NoOfTrafficlights)
             Increment NoOfTrafficlights by 1

       Otherwise
              Print "Can't add more traffic lights, maximum value reached", newline
  dropLight(droplightid)
       Assign false isfound
        Set i to 0
        Repeat while i is less than NoOfTrafficlights
                  if droplightid is equal to TrafficLight_Objects(i).getID()
                      Print "The light to be removed is found", " Light L" <<droplightid << " is
                      removed."  newline
                      Set j to i
                      Repeat while j is less than NoOfTrafficlights
                              Assign TrafficLight_Objects(j + 1) to  TrafficLight_Objects(j)
                              Increment j by 1
                      Decrement NoOfTrafficlights by 1
                      Assign true to isfound
                  Increment i by 1
```

```
If the negation of isfound is true
        Print "Can't find the traffic light to be dropped", newline

run()
 Declare elapsed_time as double
 Set countoff to 0
 Repeat while countoff is less than NoOfTrafficlights
   If TrafficLight_Objects(countoff).getstate() is equal to 0
       TrafficLight_Objects(countoff).printLightInfo(TrafficLight_Objects(countoff).
       getstate(),TrafficLight_Objects(countoff).getID())

 Repeat while 1 is true
   Assign the system time to starttime
   Call the function readTrafficData()
   Repeat
     Set i to 0
     Repeat while i is less than NoOfTrafficlights
       If TrafficLight_Objects(i).getstate() is not equal to 0
         Set k to 3
         Repeat while k is greater than 1
           TrafficLight_Objects(i).setstate(k)
           TrafficLight_Objects(i).printLightInfo(TrafficLight_Objects(i).getstate(),
           TrafficLight_Objects(i).getID())
           Set j to 0
           Repeat while j is less than NoOfTrafficlights
               If j is not equal to i and TrafficLight_Objects(j).getstate() is not equal to 0
                 TrafficLight_Objects(j).setstate(1);
                 TrafficLight_Objects(j).printLightInfo(TrafficLight_Objects(j).getstate(),
                 TrafficLight_Objects(j).getID())
                 Increment j by 1

           If k is equal to 3
             TrafficLight_Objects(i).wait(TrafficLight_Objects(i).getGreenTiming())

           If k is equal to 2
             TrafficLight_Objects(i).wait(YellowTime)

          Decrement k by 1
       Increment i by 1
     Assign instantaneous system time to end time
     Assign (endtime-starttime), in nanoseconds, to elapsed_time
   while elapsed_time is less than updateinterval*86400*10^9
```

readTrafficData()
Declare Qt as double and initialize it to 0
Create an input file stream inputfile
Open file "TrafficData" for reading as inputfile
If inputfile is in fail state
Print "Error in opening the file", newline
Otherwise
Read value from inputfile file to cyclelength
Set i to 0
Repeat While i is less than Max_Objects
Read value from inputfile to TrafficFlowRate(i)
Increment i by 1
Set k to 0
Repeat while k is less than NoOfTrafficlights
Assign Qt + TrafficFlowRate(TrafficLight_Objects(k).getID()-1) to Qt
Increment k by 1
Set j to 0
Repeat while j is less than NoOfTrafficlights
updateGreenTiming(TrafficLight_Objects(k).getID()-1) , cyclelength, Qt, j)
Increment j by 1

updateGreenTiming(Qi, C, Qt, i)
*TrafficLight_Objects(i).setGreenTiming(Qi * C / Qt)*

(Destructor)

Step 4: Implementation:

<u>Assignment4.cpp</u>

```
// This Program controls the traffic lights at the intersection of roads
// Created by : Sushil Bohara
// If any light is dropped, it means that it is completely removed but the sensor still reads
the data for corresponding roads and updates in the file



#include <iostream>
#include <chrono>
#include "TrafficLight.h"
#include "Intersection.h"
using namespace std;


int main()
{   // creating the objects of the class TrafficLight
        TrafficLight L1, L2, L3, L4, L5, L6;
        // Turn off two(arbitrary number) lights
```

```
        L2.setstate(0);
        L4.setstate(0);
        // create the object of the class "Intersection"
        Intersection I;
        // Adding 'TrafficLight' Objects
        I.AddLight(L1);
        I.AddLight(L2);
        I.AddLight(L3);
        I.AddLight(L4);
        I.AddLight(L5);
        I.AddLight(L6);
        // Print the total number of traffic lights
        cout << "Total No. of Traffic lights = " << I.getNoOfTrafficlights() << endl;
        // Drop Two(arbitrary number) Traffic lights
        I.dropLight(L1.getID());
        I.dropLight(L6.getID());
        // Print the number of traffic lights after dropping
        cout << "Total lights after dropping = " << I.getNoOfTrafficlights() << endl;
        // Run the simulation
        I.run();
        return (0);
}
```

Header File: TrafficLight.h

```
// This Header file contains the class TrafficLight with attributes, getters and setters
// It has functions to make the program wait and print the state of traffic lights
// Created by: Sushil Bohara

#pragma once
#include <iomanip>
using namespace std;
class TrafficLight {
private:
        // Declaring private attributes
        int ID;
        int state;
        double GreenTiming;
        static int NoOfTrafficlights; // static variable
public:
        // Default Constructor
        TrafficLight() {
        state = 1;
        GreenTiming = 0;   // (in seconds)
        NoOfTrafficlights++;
        ID = NoOfTrafficlights;
        }
```

```
// Non Default Constructor
TrafficLight(int newstate, double newGreenTiming) {
state = newstate;
GreenTiming = newGreenTiming;
}
// Getters and Setters
int getID() {
return ID;
}
int getstate() {
return state;
}
double getGreenTiming() {
return GreenTiming;
}
static int getNoOfTrafficlights() {
return NoOfTrafficlights;
}
void setstate(int newstate) {
state = newstate;
}
void setGreenTiming(double newGreenTiming) {
GreenTiming = newGreenTiming;
}
// Function to make the program wait for specific seconds
void wait(double seconds) {
// Declare local variable for duration to wait
double waitduration;
// find the initial system time
auto starttime = chrono::steady_clock::now();
// Loop to make the program wait..
do
{   // calculate instantaneous system time
        auto endtime = chrono::steady_clock::now();
        // Calculate the difference between instantaneous time and initial time
        waitduration =
double(chrono::duration_cast<chrono::nanoseconds>(endtime - starttime).count());

} while (waitduration < (seconds * 1e9)); // Repeat the loop until the specific
seconds(converted to nanoseconds) are passed


}
// Function to print the state of the traffic lights
void printLightInfo(int state, int ID) {
// Print Off if state is 0
```

```
        if (state == 0) {
                cout << "L" << ID << setw(15) << "Off" << endl;
        }
        // Print Red if state is 1
        if (state == 1) {
                cout << "L" << ID << setw(15) << "Red" << endl;
        }
        // Print Yellow if state is 2
        if (state == 2) {
                cout << "L" << ID << setw(15) << "Yellow" << endl;
        }
        // Print Green if state is 3
        if (state == 3) {
                cout << "L" << ID << setw(15) << "Green" << endl;
        }
        }
        // destructor
        ~TrafficLight() {

        }

};
// initialize the static variable
int TrafficLight::NoOfTrafficlights = 0;
```

<u>**Header File: Intersection.h**</u>

```
// This Header file contains the class Intersection with attributes, getters and setters
// It has functions to add/drop traffic lights, run the traffic light simulation,  read data from
the file and update the Green Timing
// Created by: Sushil Bohara

#pragma once
#include <fstream>
// Declare constants
# define YellowTime 3           // Duration of Yellow Light
# define Max_Objects 8          // Maximum no. of traffic lights
# define updateinterval 24      // Green Timing update interval
class Intersection {
        // attributes
private:
        double cyclelength;
        int NoOfTrafficlights;
        //Array of TrafficLight Objects
        TrafficLight TrafficLight_Objects[Max_Objects];
        // Array of traffic flow rates
```

```
        int TrafficFlowRate[Max_Objects];
public:
        // Default Constructor
        Intersection() {

        NoOfTrafficlights = 0;
        // Initialize cyclelength and traffic flow rates to safe values
        cyclelength = 400; // in seconds
        for (int i = 0; i < Max_Objects; i++)
        {
                TrafficFlowRate[i] = 100;
        }

        }
        // Getters and Setters
        int getNoOfTrafficlights() {
        return NoOfTrafficlights;
        }
        double getcyclelength() {
        return cyclelength;
        }
        int* getTrafficFlowRate() {
        return TrafficFlowRate;
        }
        void setcyclelength(double newcyclelength) {
        cyclelength = newcyclelength;
        }
        void setTrafficFlowRate(int newrate[]) {
        for (int i = 0; i < Max_Objects; i++)
                TrafficFlowRate[i] = newrate[i];
        }

        // Function to add a new traffic light object

        void AddLight(TrafficLight alight) {
        //add new object only if the maximum number has not been reached
        if (NoOfTrafficlights < Max_Objects) {
                TrafficLight_Objects[NoOfTrafficlights] = alight;
                NoOfTrafficlights++;
        }
        else
                cout << "Can't add more traffic lights, maximum value reached" << endl;
        }
        // Function to drop a traffic Light Object searching by ID
        void dropLight(int droplightid) {
```

```
        bool isfound = false;
        // Loop to search the ID of the traffic light to be dropped
        for (int i = 0; i < NoOfTrafficlights; i++) {
                // if ID is found,replace that traffic light object by next in the array and so
on..
                if (droplightid == TrafficLight_Objects[i].getID())
                {
                cout << "The light to be removed is found." << " Light L" << droplightid <<
" is removed." << endl;
                for (int j = i; j < NoOfTrafficlights; j++) {
                        TrafficLight_Objects[j] = TrafficLight_Objects[j + 1];
                }



                NoOfTrafficlights--;
                isfound = true; // Assign true to isfound
                }
        }
        // if ID is not found, inform the user via a message.
        if (!isfound)
                cout << "Can't find the traffic light to be dropped" << endl;
        }

        // Function to run the simulation

        void run() {

        // Declare a variable to store the time duration
        double elapsed_time;
        // if any traffic lights are turned off, display "OFF" message in the screen
        for (int countoff = 0; countoff < NoOfTrafficlights; countoff++) {
                if (TrafficLight_Objects[countoff].getstate() == 0) {

TrafficLight_Objects[countoff].printLightInfo(TrafficLight_Objects[countoff].getstate(),
TrafficLight_Objects[countoff].getID());

                }

        }
        cout << endl;

        // infite loop to run the simulation....

        while (1) {
                // find the initial system time
```

```
auto starttime = chrono::steady_clock::now();
// Read the data from the file calling another function
readTrafficData();
// Loop which runs continuously for 24 hours.
do {
// Loop that goes through all the TrafficLight objects

for (int i = 0; i < NoOfTrafficlights; i++)
{   // Loop to assign the state of the 'on' traffic light object to green(3) and
yellow(2)
        if (TrafficLight_Objects[i].getstate() != 0) {
                for (int k = 3; k > 1; k--)
                {

                // assign the state of ith traffic Light state to k
                TrafficLight_Objects[i].setstate(k);
                // print the color of ith traffic light object

TrafficLight_Objects[i].printLightInfo(TrafficLight_Objects[i].getstate(),
TrafficLight_Objects[i].getID());

                // Loop to assign the state of all other 'on' Traffic Light objects
to Red(1)
                for (int j = 0; j < NoOfTrafficlights; j++)
                {
                        if (j != i && TrafficLight_Objects[j].getstate() != 0) {
                        // Assign the state of jth Traffic Light Object to red(1)
                        TrafficLight_Objects[j].setstate(1);
                        // Print the color of the jth traffic light object

TrafficLight_Objects[j].printLightInfo(TrafficLight_Objects[j].getstate(),
TrafficLight_Objects[j].getID());
                        }
                }
                // Print a newline to make clear visibility of change of state in
the traffic light.
                cout << endl;
                // wait for the duration equal to green timing if green light is on
in ith object.
                if (k == 3)

TrafficLight_Objects[i].wait(TrafficLight_Objects[i].getGreenTiming());
                // wait for the duration equal to yellow timing if yellow light is
on in ith object.
                if (k == 2)
                        TrafficLight_Objects[i].wait(YellowTime);
```

```
                    }
                }
            }
            // Find the instantaneous system time
            auto endtime = chrono::steady_clock::now();
            // Calculate the elapsed time in nanoseconds
            elapsed_time =
double(chrono::duration_cast<chrono::nanoseconds>(endtime - starttime).count());
            } while (elapsed_time < (updateinterval * 86400 * 1e9));  // Exit the loop in 24
hours to read new data from the file

        }

        }
        // Function to read New Data from the file
        void readTrafficData() {
        // Create a variable to store the total traffic flow rate and initialize to 0
        double Qt(0);
        // create the file stream to read data from the file
        ifstream inputfile;
        inputfile.open("TrafficData.txt");
        // Dispaly the error message if the file is not opened
        if (inputfile.fail()) {
            cerr << "Error in opening the file" << endl;
        }
        // If file is opened, read the data..
        else

        { // Read the cyclelength from the first line
            inputfile >> cyclelength;
            // read the traffic flow rates of all the traffic light objects (This will not give
errors if any objects were dropped)
            for (int i = 0; i < Max_Objects; i++)
            {
            inputfile >> TrafficFlowRate[i];
            }

            // calculate the total traffic flow rate
```

```
                for (int k = 0; k < NoOfTrafficlights; k++)
                {
                // Since the ID of the object is 1 more than the index of the corresponding
traffic flow rate,
                //we can write traffic flow rate of jth object =
TrafficFlowRate[TrafficLight_Objects[j].getID()-1]
                // Doing this will not give errors if any objects were dropped
                Qt += TrafficFlowRate[TrafficLight_Objects[k].getID() - 1];

                }
                // Call the function to update the green timings of all the traffic lights
                for (int j = 0; j < NoOfTrafficlights; j++) {

                updateGreenTiming(TrafficFlowRate[TrafficLight_Objects[j].getID() - 1],
cyclelength, Qt, j);

                }


        }




        }
        // Function to update the green timings of all the traffic light objects
        void updateGreenTiming(int Qi, double C, double Qt, int i) {

        TrafficLight_Objects[i].setGreenTiming((Qi * C) / Qt); // Calculate the green timing
and update.
        }

        // destructor
        ~Intersection() {

        }

};
```

Step 5: Software Testing and Verification

Test Case 1: Trying to add more lights than Maximum value

```
Can't add more traffic lights, maximum value reached
L1          Green
L2            Red
L3            Red
L4            Red
L5            Red
L6            Red
L7            Red
L8            Red
```

Test Case 2: Dropping two Lights out of eight

```
Total lights = 8
The light to be removed is found. Light L3 is removed.
The light to be removed is found. Light L4 is removed.
Total lights after dropping = 6
```

Test Case 3: Two lights(L2 and L4) out of 4 are turned off and the program still continues

```
L2            Off
L4            Off

L1          Green
L3            Red

L1         Yellow
L3            Red

L3          Green
L1            Red

L3         Yellow
L1            Red

L1          Green
L3            Red

L1         Yellow
L3            Red

L3          Green
L1            Red
```

Test Case 4: Two lights(L2 and L3) out of 4 are dropped and the program still continues

```
The light to be removed is found. Light L2 is removed.
The light to be removed is found. Light L3 is removed.

L1          Green
L4            Red

L1         Yellow
L4            Red

L4          Green
L1            Red

L4         Yellow
L1            Red

L1          Green
L4            Red

L1         Yellow
L4            Red
```

Test Case 5: Accessing values assigned by constructors of 'TrafficLight' class with the help of getters

```
For L1: ID=1, State= 1, GreenTiming= 0
For L2: ID=2, State= 1, GreenTiming= 0
For L3: ID=3, State= 1, GreenTiming= 0
For L4: ID=4, State= 1, GreenTiming= 0
```

Test Case 6: Accessing values assigned by constructors of 'Intersection' class with the help of getters

```
Default Cyclelength = 400
Total Number of TrafficLights = 4
```

```
The Traffic Flow Rates are :
Traffic Flow Rate of R1 =100
Traffic Flow Rate of R2 =100
Traffic Flow Rate of R3 =100
Traffic Flow Rate of R4 =100
```

Test Case 7: Program running with 4 Lights

```
L1        Green
L2          Red
L3          Red
L4          Red

L1       Yellow
L2          Red
L3          Red
L4          Red

L2        Green
L1          Red
L3          Red
L4          Red

L2       Yellow
L1          Red
L3          Red
L4          Red

L3        Green
L1          Red
L2          Red
L4          Red

L3       Yellow
L1          Red
L2          Red
L4          Red

L4        Green
L1          Red
L2          Red
L3          Red

L4       Yellow
L1          Red
L2          Red
L3          Red
```

User's Guide:

This program controls the traffic light simulation at the intersection of roads. To run the program, compile the given cpp file. Also, make sure that the header files “TrafficLight.h” and “Intersection.h” are included in the program. The txt file named “TrafficData.txt” containing the cycle length and the traffic flow rates must also be kept in the project directory.