Traffic Light Control System

Sagynbek Talgatuly
ENGR-AD 1000, Fall 2020

STEP 1: Problem Identification and Statement

Problem statement is to develop a software system, which will do the following:

- Simulate the traffic lights system for an intersection of four roads with corresponding four traffic lights and continue simulating until all the lights are in off state.
- Update the data regularly by receiving flowrates and cycle length data from a file
- Set the green timings of the traffic lights depending on the data from a file
- Display the states of the traffic lights according to the green timings of each traffic light and the yellow timing which is a preset constant
- Provide the opportunity to include more than four roads per intersection and remove a light from an intersection if needed

STEP 2: Gathering of Information and Input and Output Description

This assignment involves the design of a software to control a system of traffic lights at an intersection. An intersection consists of two streets that cross at right angles. Each street has a single lane in each direction (no lanes designated for left turn). The software controls the 4 traffic lights (2 direction x 2 roads) as shown in the figure below.

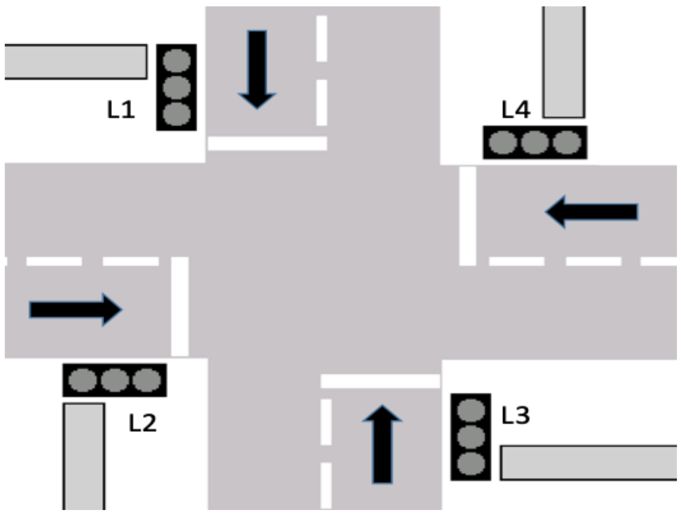


Figure 1: Intersection with traffic lights (L stands for traffic light while R stands for road).

The system has the following components:

a) Traffic semaphores (signal lights): these are standard semaphores with three lights: red, yellow, and green.
b) 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).
c) 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.

d) The green timings for the traffic lights are updated regularly based on traffic flow. The program assumes that the traffic information is stored in a file (cycle time and traffic flow rates). Every specific duration (say 24 hours), the data is read from the file, the green timings are updated based on the latest traffic condition, and the control proceeds with the updated green timings. If a traffic light turns off (for instance, if it fails), the simulation must continue with the remaining traffic lights.

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

$$di = \frac{Qi}{Qt} * C$$

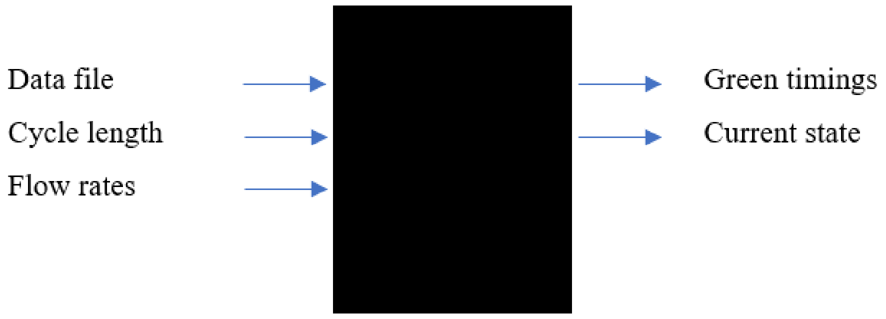
Where di is the green time for the ith traffic light, Qi represents the traffic flow (number of vehicles per hour) crossing the ith traffic light, Qt represents the total traffic flow passing through the intersection, and C represents the cycle length in seconds. It is worth note that the cycle length and traffic flow information must be read from a file (where it is assumed that the sensory data is written into). It is clear that green timings can be not integers by according to the formula. However, it is assumed that the green timings will be rounded to the nearest integer. If the calculation of the green timing leads it being 0, 1 will be set instead.

Here, it would be worth to describe the inputs/outputs of the entire program. Further in particular parts additional and more specified I/O descriptions will be included.

<u>The inputs</u>: Data file, Cycle length, Flow rates for each of the traffic lights

<u>The outputs</u>: Green timings for each of the traffic lights, The current state of the traffic lights

The black-box diagram of input/output of the program is provided below



STEP 3: Test Cases and Algorithm Design

A) Test Cases

There are many functions included in the software solution. Some of them are class functions and others are separate ones. In this step, the most general functions (that include other less significant ones in them) to the assignment ones will be tested.

1. Pausing functionality and preset yellow timing

   As stated in the problem statement, the program needs to display states of the traffic lights in accordance with the green timings. For this purpose, the program will need to pause displaying states for certain amount of time and only after this amount of time passes it will need to change/display the states. To test this functionality green time value of 21 seconds (absolutely random) and 3 seconds (yellow time) will be tested. It is expected that the program will pause displaying the states for these amounts of time in each of the cases.

2. Class TrafficLight

   In this test case, the functions of the TrafficLight class will be tested. To check whether all the functionalities properly work, the next testing algorithm should be sufficient:
   a) Print the total number of lights using the accessor for the static data member
   b) Instantiate four objects of this class and repeat the step a)
   c) Print state of the first object by using the corresponding accessor
   d) Print greenTime of the second object by using the corresponding accessor
   e) Modify the state of the first object by using the corresponding modifier
   f) Modify the greenTime of the second object by using the corresponding modifier
   g) Repeat the steps d) and e)
   h) Print the ID of the third object by using the corresponding accessor
   i) Call the printInfoFunction for all the existing objects

   The desription of the expected output is below:
   a) Total number of the lights at the beginning is 0
   b) After 4 objects are created the total number of the lights is 4
   c) state after the initialization is 1
   d) greenTime after the initialization is 0
   e) state is modified to 2
   f) greenTime is modified to 40
   g) Numbers 2 and 40 are printed
   h) Number 3 is printed (because the IDs are set by incrementing it each time new object is instantiated)
   i) ID is 1, state is 2, greenTime is 0 (since it wasn't modified)
      ID is 2, state is 1 (since it wasn't modified), greenTime is 40
      ID is 3, state is 1 (since it wasn't modified), greenTime is 0 (since it wasn't modified)
      ID is 4, state is 1 (since it wasn't modified), greenTime is 0 (since it wasn't modified)

3. Class Intersection

   In this test case, the functions of the Intersection class will be tested. To check whether all the functionalities properly work, the next testing algorithm should be sufficient:
   a) Print the number of lights stored in the intersection using the accessor for it
   b) Instantiate five lights and add them to the intersection object and repeat a)
   c) Print cycleLength by using the corresponding accessor
   d) Print the flowRates[] values

   The desription of the expected output is below:
   a) Number of the lights at the beginning is 0

b) After 5 objects are added the number of the lights stored in the array is 5
c) CycleLength at the initialization os 120
d) Default flowRates[] values are 1000

4. Functions addLight and dropLight of the class Intersection

To test if addLight functions work, 6 different traffic light objects will be created and will be added to the intersection object. Conducting the validation in the following way should be sufficient:

a. Create two intersection objects
b. Create 6 traffic light objects
c. Add three of them to the first intersection
d. Add the other three of them to the second intersection
e. Call the run function to them separately
f. Drop one traffic light from each of the intersections
g. Repeat the step e)

The expected result is that in the first two simualation there will be three traffic lights in each of them, and when one traffic light is dropped from each of the intersections the simulations will run for two traffic lights.

5. Updating functionality

In this test case the updating functionality will be tested. To do this, the update interval will be set to different values: 10, 30, 60. It is expected that in each case the data will be updated regularly every time 10, 30, 60 seconds pass.

6. Simulation

In this test case all the mentioned above functions will be tested in the simulation itself together. In each of the test cases it is expected that the update functionalities, green time calculations, and pausing functionalities are conducted properly. The test cases here can be divided as follows, (it is assumed that the cycleLength provided is an integer and flow rates provided are positive):

Test case 1: double values for flow rates (the cycle length is 60 seconds)
For the flow rates values, green time values can be calculated by hand. The expected calculation are provided below.

Flow rates	***Green time***
776.25	*14*
560.50	*10*
988.55	*18*
995.67	*18*

Test case 2: integer values for flow rates (the cycle length is 45 seconds)
For the flow rates values, green time values can be calculated by hand. The expected calculation are provided below.

Flow rates	***Greem time***
700	*9*
900	*11*
800	*10*
1200	*15*

Test case 3: extreme values for flow rates (the cycle length is 30 seconds)
For the flow rates values, green time values can be calculated by hand. The expected calculation are provided below.

Flow rates	***Green time***
0	*1*
1	*1*
1000000	*15*
1000000	*15*

Test case 4: more than 4 traffic lights (the cycle length is 60 seconds)
For the flow rates values, green time values can be calculated by hand. The expected calculation are provided below.

Flow rates	***Green time***
1000	*6*
2000	*13*
1500	*10*
1200	*8*
800	*5*
900	*6*
2000	*13*

Test case 5: some of the lights are turned off
In this test case, seven lights will be added into the intersection. Two of them will be turned off. It is expected that during the simulation these lights will be simply skipped by the program and no time for them will be allocated. Instead, the time for turned off lights will be distributed to other green timings. The flow rates and cycle length value are from the test case number 4. For the flow rates values, green time values can be calculated by hand. The expected calculation are provided below.

Flow rates	***Green time***
1000	*10*
2000	*0*
1500	*15*
1200	*0*
800	*8*
900	*9*
2000	*19*

Test case 6: all the lights are turned off

In this case, it is assumed that the program will be terminated if all the lights are turned off.

B) Algorithm (pseudocode)

```
Function wait
    Pass In: t
    Assign current time to a
    Assign current time – a to c
    Print “waiting… ”
```

```
        Repeat the following while current time is less than (a+t)
            If current time – a is more than c then
                Assign 0 to j
                Assign the greatest integer that is less or equal to logc/log10 to k
                Repeat the following while j is less than k + 1
                    Print "\b \b"
                    Increment j by 1
                Assign current time – a to c
                Print c
        If c is not 0 then
            Assign 0 to j
            Assign the greatest integer that is less or equal to logc/log10 to k
            Repeat the following while j is less than k + 12
                    Print "\b \b"
                    Increment j by 1
            Otherwise assign 0 to i
                Repeat the following while i is less than 13
                    Print "\b \b"
                    Increment i by 1
        Pass Out: nothing
    Function stateConverter
        Pass In: state
        If state equals 0 then
            Assign "Off" to out
            Otherwise if state equals 1 then
                Assign "Red" to out
                Otherwise if state equals 2 then
                    Assign "Yellow" to out
                    Otherwise if state equals 3 then
                        Assign "Green" to out
                        Otherwise assign "Red" to out
        Pass Out: out
    Define class TrafficLight
        Private members
            id as integer
            state as integer
            greenTime as integer
            totalNumberOfLights as static integer
        Public members
            Constructor TrafficLight
                Pass In: nothing
                Increment totalNumberOfLights by 1
                Assign totalNumberOfLights to id
                Assign 1 to state
                Assign 0 to greenTime
            Accessor getID
                Pass In: nothing
                Pass Out: id
            Accessor getState
                Pass In: nothing
                Pass Out: state
            Accessor getGreenTime
                Pass In: nothing
                Pass Out: greenTime
                Accessor getTotalNumOfLights
                        Pass In: nothing
                        Pass Out: totalNumberOfLights
                Modifier setState
                        Pass In: stateVal
                        Assign stateVal to state
                        Pass Out: nothing
                Modifier setGreenTime
                        Pass In: greenTimeVal
                        Assign greenTimeVal to greenTime
                        Pass Out: nothing
                Function printLightInfo
                        Pass In: nothing
                        Print “ID: ”, id
                        Print “State: ”, state
                        Print “GreenTiming: ”, greenTime
                        Pass Out: nothing
Initialize a static variable totalNumberOfLights to 0
Define MaxNumberOfLights as 20
Define class Intersection
        Private members
                cycleLength as integer
                numOfLights as integer
                Lights as array of type TrafficLight with size MaxNumberOfLights
                flowRates as array of type double with size MaxNumberOfLights
        Public members
                Constructor Intersection
                        Pass In: nothing
                        Assign 0 to numOfLights
                        Assign 120 to cycleLength
                        Assign 0 to j
                        Repeat the following while j is less than MaxNumberOfLights
                                Assign 1000 to flowRates[j]
                                Increment j by 1
                Accessor getCycleLength
                        Pass In: nothing
                        Pass Out: numOfLights
                Accessor getNumOfLights
                        Pass In: nothing
                        Pass Out: numOfLights
                Accessor getFlowRates
                        Pass In: index
                        Pass Out: flowRates[index]
                Function addLight
                        Pass In: Traffic Light object A
                        If numOfLights is more or equal to MaxNumberOfLights then
                                Print “Cannot add more traffic lights”
                                Otherwise assign A to Lights[numOfLights]
                                        Increment numOfLights by 1
                        Pass Out: nothing
                Function dropLight
                        Pass In: Traffic Light object A
                        Call getID function of the object A and assign its return value to keyID
                        Assign -1 to foundIndex
        Assign 0 to j
        Repeat the following while j is less than numOfLights
            Call getID function of the object Lights[j] and assign its return value to k
            If keyID is equal to k then
                Assign j to foundIndex
                Exit the loop
            Increment j by 1
        If foundIndex is -1 then
            Assign false to Output
            Otherwise decrement numOfLights by 1
                Assign foundIndex to i
                Repeat the following while i is less than numOfLights
                    Assign Lights[i+1] to Lights[i]
                    Increment i by 1
                Assign true to Output
        Pass Out: Output
    Function readTrafficData
        Pass In: nothing
        Initialize an input sensor and open “SensorData.txt” with it
        If input sensor fails, then
            print error message “File could not be opened”
            exit the program
        Read the cycle length value from the input file with an input sensor and assign it
            to cycleLength
        Assign 0 to j
        Repeat the following while j is less than numOfLights
            Read the flow rate value from the input file with an input sensor and
                assign it to flowRate[j]
            Increment j by 1
        Close the input sensor
        Pass Out: nothing
    Function updateTiming
        Pass In: nothing
        Call readTrafficData function
        Assign 0 to j
        Repeat the following while j is less than numOfLights
            Call getState function of the object Lights[j] and if its return is 0 then
                Assign 0 to flowRates[j]
            Increment j by 1
        Assign 0 to sum
        Assign 0 to j
        Repeat the following while j is less than numOfLights
            Increment sum by flowRates[j]
            Increment j by 1
        Assign 0 to p
        Repeat the following while p is less than numOfLights
            Assign cycleLength * flowRates[p] / sum to calculatedTiming
            Round calculatedTiming to the nearest integer
            Call getState function of the object Lights[j] and assign its return to st
            If calculatedTiming is 0 and st is not equal to 0 , then
                Call setGreenTime function of the Lights[p], passing 1
                Otherwise call setGreenTime function of the Lights[p], passing
                    calculatedTiming
            Increment p by 1
        Pass Out: nothing
Function announceTheUpdate
        Pass In: nothing
        Print “-Cycle length and flow rates were read by the program successfully”
        Print “-Timings for the green state was updated and provided below”
        Print “-Please, note that the values were rounded”
        Print “Cycle length is: ”
        Assign 0 to j
        Repeat the following while j is less than numOfLights
                Print “Green timing for a traffic light with an ID ”
                Call getID function of the Lights[j] and print its return value
                Print “is: ”
                Call getGreenTime function of the Lights[j] and print its return value
                Increment j by 1
        Pass Out: nothing
Function checkAllTheLights
        Pass In: nothing
        Assign flag to 0
        Assign 0 to j
        Repeat the following while j is less than numOfLights
                Call getState function of the Lights[j] and increment flag by its return
                        value
                Increment j by 1
        If flag is 0 then
                Print “--> The program stopped because all the lights are off!”
                Exit the program
        Pass Out: nothing
Function updateIntervalChecker
        Pass In: initTime, updateInterval
        If current time is more or equal to initTime + updateInterval then
                Call updateTiming function
                Assign current time to initTime
                Print updateInterval, “seconds have passed since the last update”
                Call announceTheUpdate function
                Assign 0 to j
                Repeat the following while j is less than numOfLights
                Print “Light ”
                Call getID function of the Lights[j] and print its return value
                Increment j by 1
        Pass Out: nothing
Function run
        Pass In: yellowTiming, updateInterval
        Print “Preset timing for the yellow state is ”, yellowTiming, “ seconds”
        Print “Preset interval for the updates is ”, updateInterval, “ seconds”
        Call updateTiming function
        Call announceTheUpdate function
        Assign current time to initTime
        Assign 0 to j
        Repeat the following while j is less than numOfLights
                Print “Light ”
                Call getID function of the Lights[j] and print its return value
                Increment j by 1
        Repeat the following while 1 equals 1
                Call checkTheLights function
```

```
                                        Assign 0 to j
                                        Repeat the following while j is less than numOfLights
                                                Call getState function of the Lights[j] and assign its return to K
                                                If K is 0 then skip the iteration
                                                Otherwise call setState function of the Lights[j], passing 3
                                                        Assign 0 to p
                                                        Repeat the following while p is less than numOfLights
                                                                Call stateConverter function, passing Lights[p]
                                                                        and print its return value
                                                                Increment p by 1
                                                        Call wait function, passing K
                                                        Call updateIntervalChecker function, passing initTime,
                                                                updateInterval
                                                        Call setState function of the Lights[j], passing 2
                                                        Repeat the following while p is less than numOfLights
                                                                Call stateConverter function, passing Lights[p]
                                                                        and print its return value
                                                                Increment p by 1
                                                        Call wait function, passing yellowTiming
                                                        Call updateIntervalChecker function, passing initTime,
                                                                updateInterval
                                                        Call setState function of the Lights[j], passing 1
                                                Increment j by 1
                                Pass Out: nothing
The main function
        Instantiate an object L1 of class TrafficLight
        Instantiate an object L2 of class TrafficLight
        Instantiate an object L3 of class TrafficLight
        Instantiate an object L4 of class TrafficLight
        Instantiate an object I1 of class Intersection
        Call addLight function of the object I1, passing L1
        Call addLight function of the object I1, passing L2
        Call addLight function of the object I1, passing L3
        Call addLight function of the object I1, passing L4
        Assign 3 to yellow timing
        Assign 86400 to updateInterval
        Call run function of the object I1, passing yellowTiming, updateInterval
        Exit
```

STEP 4: Implementation

Source.cpp

```
/*-------------------------------------------------------------------------*/
/* Name: Sagynbek Talgatuly Student Number: st4121                        */
/* Date: November 27. 2020                                                */
/* Program: Source.cpp                                                    */
/* Description: This program simulates a traffic lights system for the    */
/*              intersection of four roads and four lights respectively   */
/*              according to the data provided in the sensor file.        */
/*-------------------------------------------------------------------------*/

//including all the needed libraries
#include <iostream>
#include <ctime>
#include <cmath>
#include <fstream>

using namespace std;

//think of a way to include the wait function inside the TrafficLight class

//the wait function is used to pause the program and let it
//continue only when certain amount of time passes
void wait(int t)
{
	//initial time is assigned to a variable
	long long int a = time(NULL);
	//current time is assigned to a variable
	//it is assumed that at this assignment moment the c variable is 0
	long long int c = time(NULL) - a;
	//next it print the phrase waiting...
	cout << "waiting... ";
	//until the required amount of time has not passed, the program does the following
	while (time(NULL) < (a + t)){
		//it checks if the current time changed
		if (time(NULL) - a > c) {
			//and if it changed, it deletes the printed time and displays the current
			//time in other words it just increments the current time displayed
			//to delete a character, \b delimiter was used
			//to find out how many times we need to delete, logarithm of 10 was used
			for (int i = 0; i < (int)(log(c) / log(10)) + 1; i++) { cout << "\b \b"; }
			//the current time is set
			c = time(NULL) - a;
			//and printed
			cout << c;
		}
	}
	//after the required time passes, the program removes everything it printed
	//again, depending on how many digits the current time has, the \b is used
	//it also removes the word 'waiting... ', that is why we need to delete 11 more times
	if (c!=0)
	for (int i = 0; i < (int)(log(c) / log(10)) + 12; i++) { cout << "\b \b"; }
	else
	for (int i = 0; i < 13; i++) { cout << "\b \b"; }
}

//the function below is used to convert the state value from
//the indication number to its meaning
string stateConverter(int state)
{
	//for this purpose switch operand was used
	switch (state)
	{
	case 0: return "Off"; break;
	case 1: return "Red"; break;
	case 2: return "Yellow"; break;
	case 3: return "Green"; break;
	default: return "Red";
	}
}

//the header files are included only at this point of the code
//because the functions wait and stateConverter are also called in these header files
#include "TrafficLight.h"
#include "Intersection.h"

int main()
```

```
{
	//here, 4 traffic light objects and one intersection object were created
	TrafficLight L1;
	TrafficLight L2;
	TrafficLight L3;
	TrafficLight L4;
	Intersection I1;
	//the traffic light objects were added to the intersection
	I1.addLight(L1);
	I1.addLight(L2);
	I1.addLight(L3);
	I1.addLight(L4);
	//the simulation for the intersection starts here
	//the yellow timing and update interval values are set before the
	//simulation and passed to the run() function
	const int yellowTiming = 3;
	const long long int updateInterval = 60*60*24;

	I1.run(yellowTiming, updateInterval);

	return 0;
}
```

Intersection.h

```
#pragma once

//the maximum number of traffic lights this class can store is 20
#define MaxNumberOfLights 20

class Intersection {
private:
	//private data members
	int cycleLength;
	int numOfLights;
	TrafficLight Lights[MaxNumberOfLights];
	double flowRates[MaxNumberOfLights];
public:
	//constructor
	Intersection()
	{
		numOfLights = 0;
		cycleLength = 120;
		for (int i = 0; i < MaxNumberOfLights; i++)
		{
			flowRates[i] = 1000;
		}
	}
	//Accessors
	int getCycleLength() { return cycleLength; }
	int getNumOfLights() { return numOfLights; }
	int getFlowRates(int index) { return flowRates[index]; }
	//function to add new traffic lights to the class
	void addLight(TrafficLight A) {
		//if the current number of lights is at the boundary,
		//the message that no more additions are possible displayed
		if (numOfLights >= MaxNumberOfLights)
			cout << "Cannot add more traffic lights" << endl;
		else
		{
			//if the number of lights is ok, then the object is added to the
			//array of traffic lights and number of traffic lights
			//in this particlar intersection is incremented
			Lights[numOfLights] = A;
			numOfLights++;
		}
	}
	//this function is used to remove a light from the intersection
	bool dropLight(TrafficLight A) {
		//the ID of the traffic light that needs to be removed is assigned
		//to some variable
		int keyID = A.getID();
		int foundIndex = -1;
		for (int i = 0; i < numOfLights; i++)
		{
			//when the light is found, its position in the array is also saved
			if (keyID == Lights[i].getID()) { foundIndex = i; break; }
		}
		//if the object wasn't found in the array, then the function returns false
		if (foundIndex == -1) return false;
```

```
        else {
                //otherwise, it removes the traffic light from the intersection
                //and moves all the elements of the array after the found element
                //one place to the left and decrements the number fo lights
                //stored in the intersection object
                numOfLights--;
                for (int k = foundIndex; k < numOfLights; k++)
                {
                        Lights[k] = Lights[k + 1];
                }
        }
        return true;
}
//this function reads the data from a file and stores it in the
//corresponding data members
void readTrafficData() {
        ifstream inFile;
        inFile.open("SensorData.txt");
        if (inFile.fail()) //open failed
        {
                cerr << "File sensor1.txt could not be opened";
                exit(1); //end execution of the program
        }
        inFile >> cycleLength;
        for (int i = 0; i < numOfLights; i++) {
                inFile >> flowRates[i];
        }
        inFile.close();
}
//this function updates the data members by calling the read function
//and also updates the greenTime data members of separate traffic lights
//included in the intersection by using the provided formula
void updateTiming() {
        readTrafficData();
        for (int i = 0; i < numOfLights; i++)
        {
                if (Lights[i].getState() == 0) { flowRates[i] = 0; }
        }
        double sum = 0;
        for (int i = 0; i < numOfLights; i++) {
                sum += flowRates[i];
        }
        for (int i = 0; i < numOfLights; i++) {
                double calculatedTiming = round(cycleLength * flowRates[i] / sum);
                if ((int)calculatedTiming == 0 && Lights[i].getState() != 0)
                        Lights[i].setGreenTime(1);
                else Lights[i].setGreenTime((int)calculatedTiming);
        }
}
//this function prints the relevant information about the update
//this function should be called only after calling the updateTiming function
void announceTheUpdate() {
        cout << endl;
        cout << "-Cycle length and flow rates were read by the program successfully" <<
                endl;
        cout << "-Timings for the green state was updated and provided below" << endl;
        cout << "-Please, note that the values were rounded" << endl;
        cout << "Cycle length is: " << cycleLength << endl;
        for (int i = 0; i < numOfLights; i++)
        {
                cout << "Green timing for a traffic light with an ID ";
                cout << Lights[i].getID() << " is: " << Lights[i].getGreenTime() << endl;
        }
        cout << endl;
}
//this function checks if any of the traffic lights is working
//in case all the traffic lights are in state 0, the program is exitted
void checkTheLights() {
        int sum = 0;
        for (int i = 0; i < numOfLights; i++)
                sum += Lights[i].getState();
        if (sum == 0)
        {
                cerr << "\n --> The program stopped because all the lights are off! \n";
                        exit(1);
        }
}
//this function checks if the update interval state at the beginning of the simulation
        has passed
//and if so it calls the update function and set the init time to the current time
```

```
        //so that it could be in the same way in the next update iterations
        void updateIntervalChecker(long long int& initTime, long long int updateInterval) {
                if (time(NULL) >= initTime + updateInterval) {
                        updateTiming();
                        initTime = time(NULL);
                        cout << endl;
                        cout << "-" << updateInterval << " seconds have passed since the last
                                update" << endl;
                        announceTheUpdate();
                        for (int k = 0; k < numOfLights; k++)
                        {
                                cout << "Light " << Lights[k].getID() << " \t";
                        }
                        cout << endl;
                }
        }
        //this is the core function for the simulatio of the traffic lights
        //system for a particular intersection
        //the yellow timing and update interval values are passed to the run() function
        //wherever this funtion is called
        void run(int yellowTiming, long long int updateInterval) {
                cout << "Preset timing for the yellow state is " << yellowTiming << " seconds" <<
                        endl;
                cout << "Preset interval for the updates is " << updateInterval << " seconds" <<
                        endl;
                //timing is update before the program starts simulating
                updateTiming();
                announceTheUpdate();
                //initial time is stored for further usages
                long long int initTime = time(NULL);
                //columns for the traffic lights are displayed
                for (int i = 0; i < numOfLights; i++)
                {
                        cout << "Light " << Lights[i].getID() << " \t";
                }
                cout << endl;
                //the following loop continues until all the lights are turned off
                while (true) {
                        //check if not all the lights are turned off
                        checkTheLights();
                        //there is a certain algorithm that goes for each of the traffic lights
                        for (int i = 0; i < numOfLights; i++) {
                                //if the light is turned off, the loop is skipped
                                //because there is no need to wait for this state
                                if (Lights[i].getState() == 0) continue;
                                //set the state to green
                                Lights[i].setState(3);
                                for (int j = 0; j < numOfLights; j++)
                                {
                                cout << stateConverter(Lights[j].getState()) << "\t\t"; }
                                cout << endl;
                                //and pause displaying for time that is equal to the green timing
                                        of this particulat traffic light
                                wait(Lights[i].getGreenTime());
                                //when the time for the green state passed, we check if the update
                                        interval passed too or not
                                //if so, we update the data
                                updateIntervalChecker(initTime, updateInterval);
                                //set the state to yellow
                                Lights[i].setState(2);
                                for (int j = 0; j < numOfLights; j++)
                                {
                                cout << stateConverter(Lights[j].getState()) << "\t\t"; }
                                cout << endl;
                                //and pause displaying for time that is equal to yellow timing
                                wait(yellowTiming);
                                //when the time for the yellow state passed, we check if the update
                                        interval passed too or not
                                //if so, we update the data
                                updateIntervalChecker(initTime, updateInterval);
                                //last, we set the state to red
                                //and the loop goes for the next light and so on untill all of them
                                        are turned off
                                Lights[i].setState(1);
                        }
                }
        }
};
```

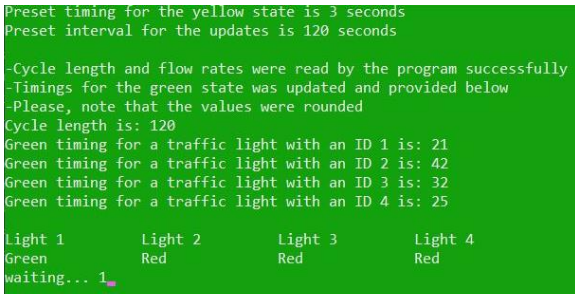
TrafficLight.h

```cpp
#pragma once

class TrafficLight {
private:
	//private data members
	int id;
	int state;
	int greenTime;
	//total number of lights is declared as a static variable
	//because it is common to the entire class
	static int totalNumberOfLights;
public:
	//default constructor
	TrafficLight()
	{
		totalNumberOfLights++;
		id = totalNumberOfLights;
		state = 1;
		greenTime = 0;
	}
	//accessors to the data members
	int getID() { return id; }
	int getState() { return state; }
	int getGreenTime() { return greenTime; }
	//modifiers of the data members
	void setState(int state) { this->state = state; }
	void setGreenTime(int greenTime) { this->greenTime = greenTime; }
	//class function to display all the data
	//members about this particular traffic light
	void printLightInfo()
	{
		cout << "ID: " << id << endl;
		cout << "State: " << state << endl;
		cout << "Green Timing: " << greenTime << endl;
	}
	//accessor to the total number of traffic light objects created
	static int getTotalNumOfLights() { return totalNumberOfLights; }
};

//the static variable is initialized
int TrafficLight::totalNumberOfLights = 0;
```

STEP 5: Tests and Verification (and Debugging)

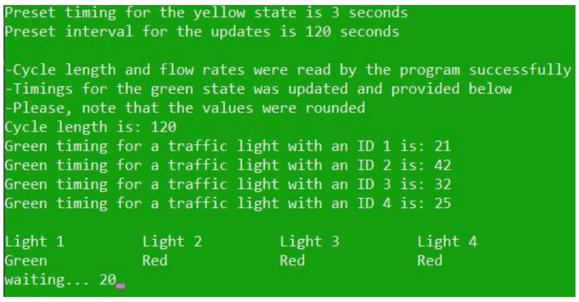
1. For the pausing tests

Test case 1: pausing interval is 21 seconds



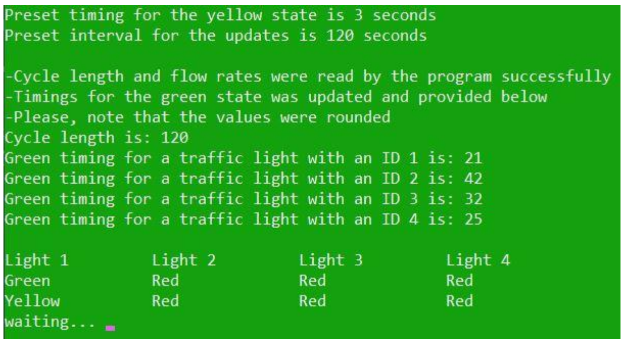
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 120
Green timing for a traffic light with an ID 1 is: 21
Green timing for a traffic light with an ID 2 is: 42
Green timing for a traffic light with an ID 3 is: 32
Green timing for a traffic light with an ID 4 is: 25

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
waiting... 1
```

(after 1 second) ^



```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 120
Green timing for a traffic light with an ID 1 is: 21
Green timing for a traffic light with an ID 2 is: 42
Green timing for a traffic light with an ID 3 is: 32
Green timing for a traffic light with an ID 4 is: 25

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
waiting... 20
```

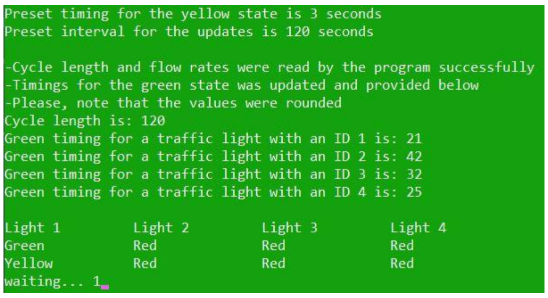
(after 20 seconds) ^



```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 120
Green timing for a traffic light with an ID 1 is: 21
Green timing for a traffic light with an ID 2 is: 42
Green timing for a traffic light with an ID 3 is: 32
Green timing for a traffic light with an ID 4 is: 25

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red
waiting...
```

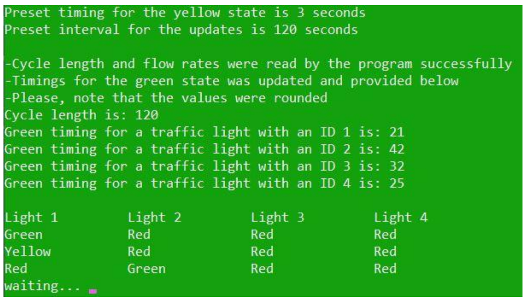
(after 21 seconds) ^

Test case 2: pausing time interval is 3 seconds



```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 120
Green timing for a traffic light with an ID 1 is: 21
Green timing for a traffic light with an ID 2 is: 42
Green timing for a traffic light with an ID 3 is: 32
Green timing for a traffic light with an ID 4 is: 25

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red
waiting... 1
```

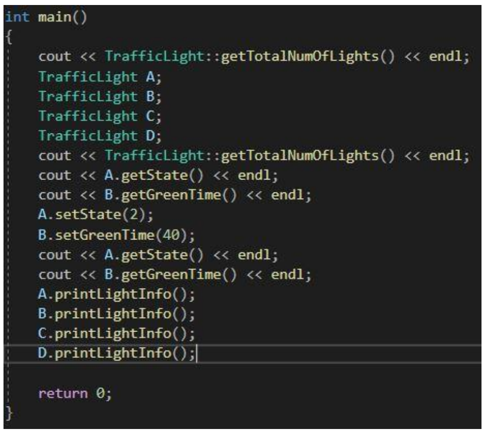
(after 1 second) ^



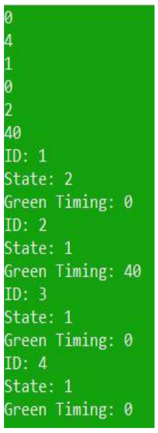
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 120
Green timing for a traffic light with an ID 1 is: 21
Green timing for a traffic light with an ID 2 is: 42
Green timing for a traffic light with an ID 3 is: 32
Green timing for a traffic light with an ID 4 is: 25

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red
Red             Green           Red             Red
waiting...
```

(after 3 seconds) ^

2. For the class TrafficLight tests

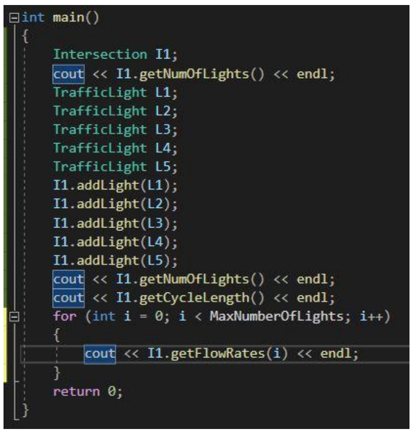


```cpp
int main()
{
    cout << TrafficLight::getTotalNumOfLights() << endl;
    TrafficLight A;
    TrafficLight B;
    TrafficLight C;
    TrafficLight D;
    cout << TrafficLight::getTotalNumOfLights() << endl;
    cout << A.getState() << endl;
    cout << B.getGreenTime() << endl;
    A.setState(2);
    B.setGreenTime(40);
    cout << A.getState() << endl;
    cout << B.getGreenTime() << endl;
    A.printLightInfo();
    B.printLightInfo();
    C.printLightInfo();
    D.printLightInfo();

    return 0;
}
```

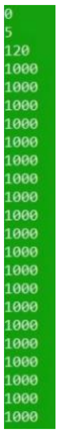


```
0
4
1
0
2
40
ID: 1
State: 2
Green Timing: 0
ID: 2
State: 1
Green Timing: 40
ID: 3
State: 1
Green Timing: 0
ID: 4
State: 1
Green Timing: 0
```

3. For the class Intersection tests

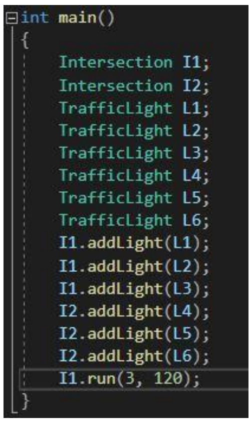


```cpp
int main()
{
    Intersection I1;
    cout << I1.getNumOfLights() << endl;
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I1.addLight(L4);
    I1.addLight(L5);
    cout << I1.getNumOfLights() << endl;
    cout << I1.getCycleLength() << endl;
    for (int i = 0; i < MaxNumberOfLights; i++)
    {
        cout << I1.getFlowRates(i) << endl;
    }
    return 0;
}
```

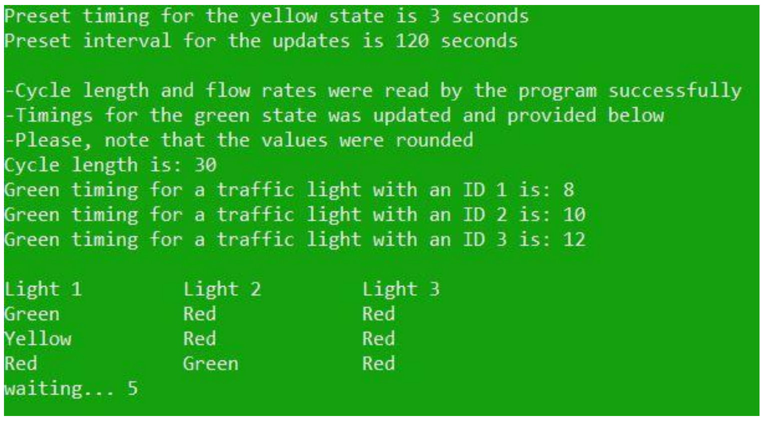


```
0
5
120
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
1000
```

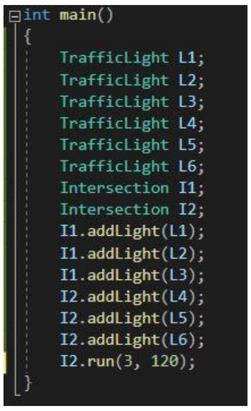
4. For the addLight and dropLight functionalities tests



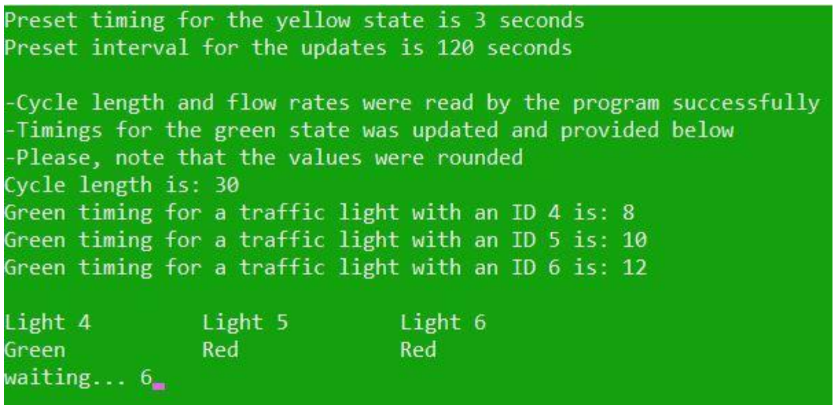
```cpp
int main()
{
    Intersection I1;
    Intersection I2;
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    TrafficLight L6;
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I2.addLight(L4);
    I2.addLight(L5);
    I2.addLight(L6);
    I1.run(3, 120);
}
```



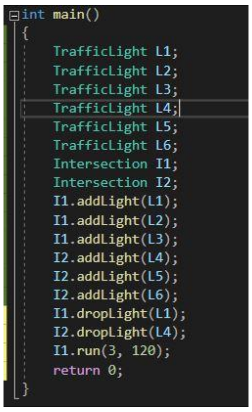
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 1 is: 8
Green timing for a traffic light with an ID 2 is: 10
Green timing for a traffic light with an ID 3 is: 12

Light 1         Light 2         Light 3
Green           Red             Red
Yellow          Red             Red
Red             Green           Red
waiting... 5
```



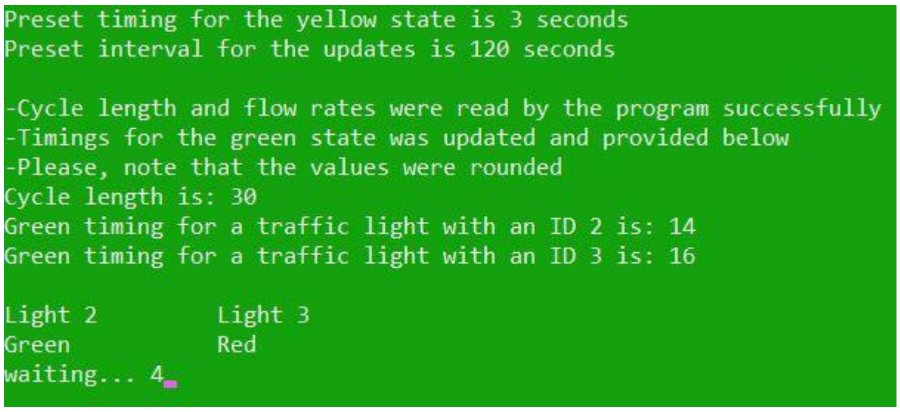
```cpp
int main()
{
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    TrafficLight L6;
    Intersection I1;
    Intersection I2;
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I2.addLight(L4);
    I2.addLight(L5);
    I2.addLight(L6);
    I2.run(3, 120);
}
```



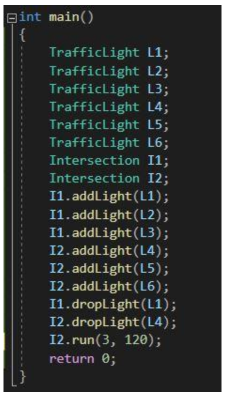
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 4 is: 8
Green timing for a traffic light with an ID 5 is: 10
Green timing for a traffic light with an ID 6 is: 12

Light 4         Light 5         Light 6
Green           Red             Red
waiting... 6
```



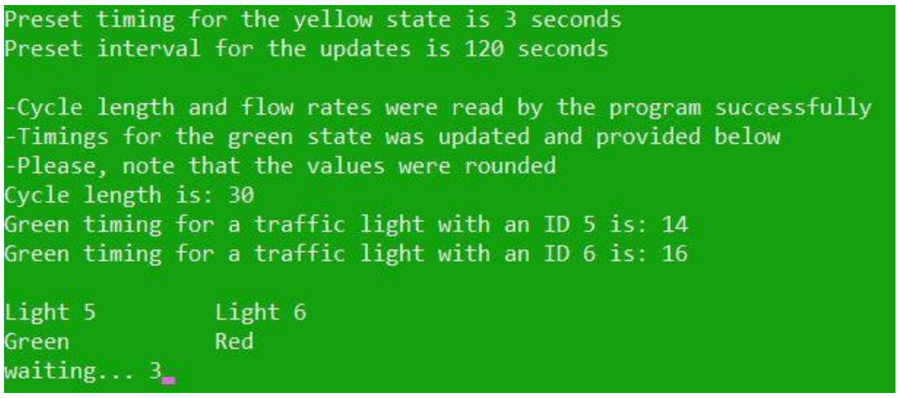
```cpp
int main()
{
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    TrafficLight L6;
    Intersection I1;
    Intersection I2;
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I2.addLight(L4);
    I2.addLight(L5);
    I2.addLight(L6);
    I1.dropLight(L1);
    I2.dropLight(L4);
    I1.run(3, 120);
    return 0;
}
```



```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 2 is: 14
Green timing for a traffic light with an ID 3 is: 16

Light 2         Light 3
Green           Red
waiting... 4
```

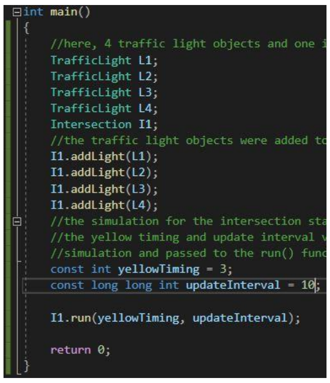


```cpp
int main()
{
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    TrafficLight L6;
    Intersection I1;
    Intersection I2;
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I2.addLight(L4);
    I2.addLight(L5);
    I2.addLight(L6);
    I1.dropLight(L1);
    I2.dropLight(L4);
    I2.run(3, 120);
    return 0;
}
```

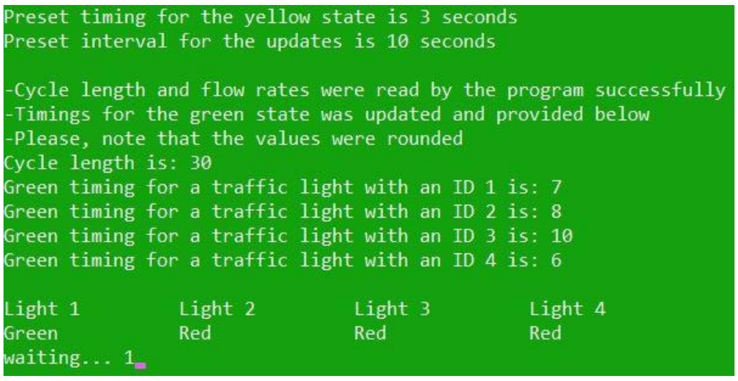


```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 5 is: 14
Green timing for a traffic light with an ID 6 is: 16

Light 5         Light 6
Green           Red
waiting... 3
```

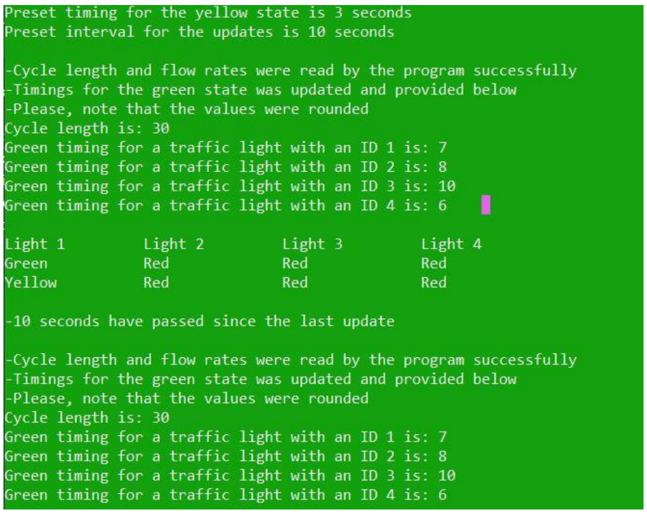
5. For the Updating functionality tests
Test Case 1: update interval = 10 seconds



```
int main()
{
    //here, 4 traffic light objects and one i
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    Intersection I1;
    //the traffic light objects were added to
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I1.addLight(L4);
    //the simulation for the intersection sta
    //the yellow timing and update interval v
    //simulation and passed to the run() func
    const int yellowTiming = 3;
    const long long int updateInterval = 10;

    I1.run(yellowTiming, updateInterval);

    return 0;
}
```



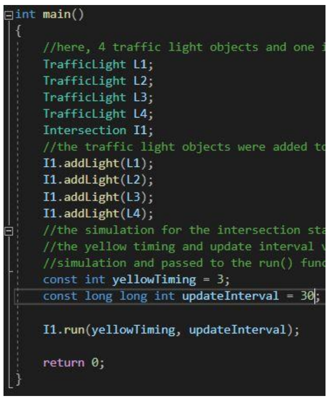
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 10 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 1 is: 7
Green timing for a traffic light with an ID 2 is: 8
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 6

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
waiting... 1
```



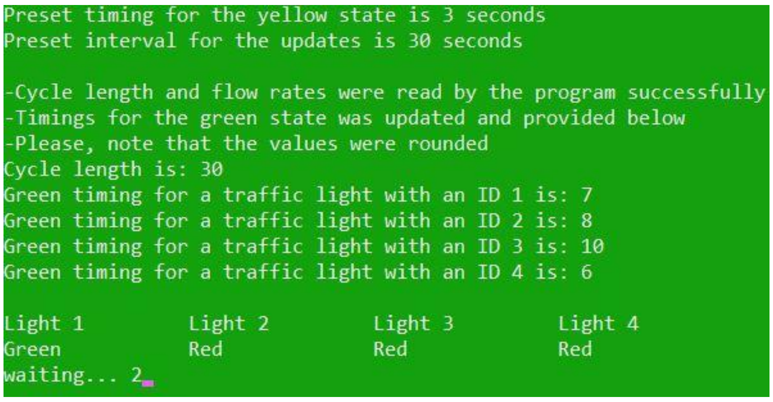
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 10 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 1 is: 7
Green timing for a traffic light with an ID 2 is: 8
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 6

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red

-10 seconds have passed since the last update

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 1 is: 7
Green timing for a traffic light with an ID 2 is: 8
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 6
```

When 10 seconds passed, the data was updated. The state was displayed with old parameters, and only after that it displayed the update message and then it followed the new parameters

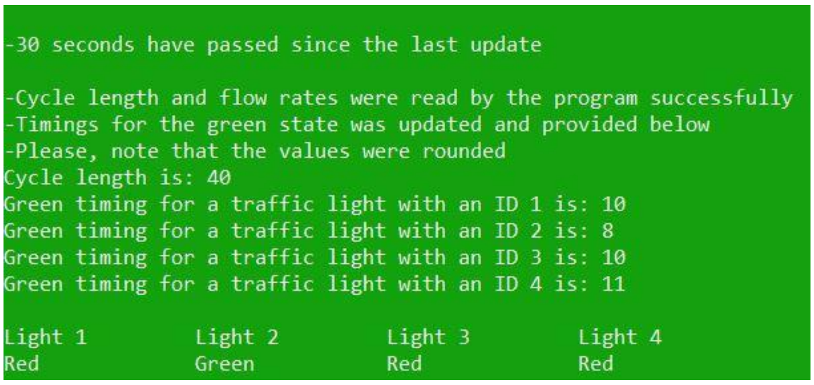
Test Case 2: update interval = 30 seconds



```
int main()
{
    //here, 4 traffic light objects and one i
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    Intersection I1;
    //the traffic light objects were added to
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I1.addLight(L4);
    //the simulation for the intersection sta
    //the yellow timing and update interval v
    //simulation and passed to the run() func
    const int yellowTiming = 3;
    const long long int updateInterval = 30;

    I1.run(yellowTiming, updateInterval);

    return 0;
}
```



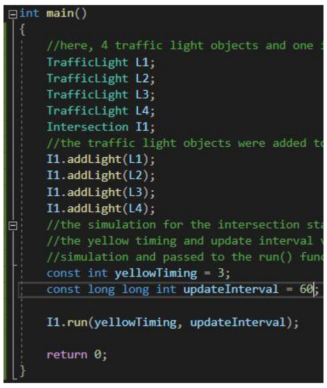
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 30 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 1 is: 7
Green timing for a traffic light with an ID 2 is: 8
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 6

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
waiting... 2
```



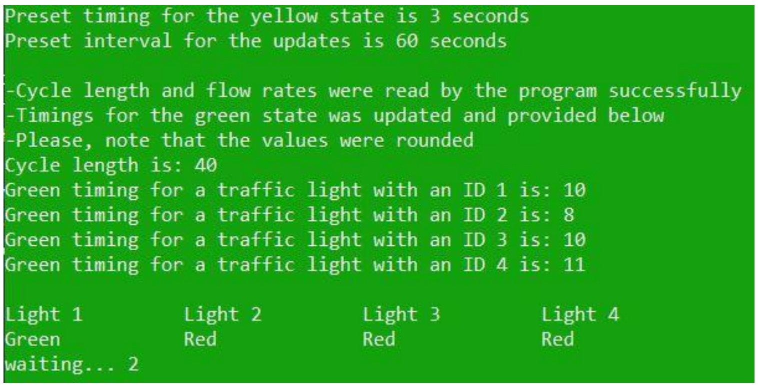
```
-30 seconds have passed since the last update

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 40
Green timing for a traffic light with an ID 1 is: 10
Green timing for a traffic light with an ID 2 is: 8
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 11

Light 1         Light 2         Light 3         Light 4
Red             Green           Red             Red
```

When 30 seconds passed, the data was updated. The state was displayed with old parameters, and only after that it displayed the update message and then it followed the new parameters

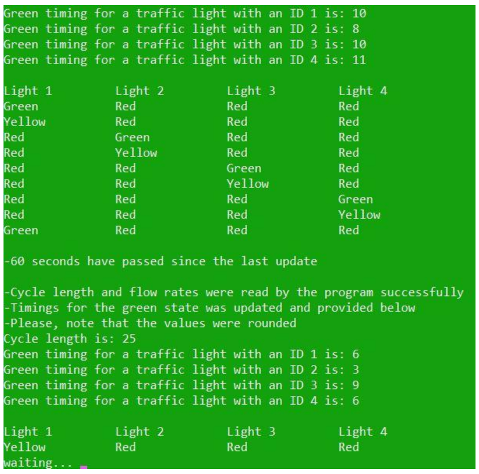
Test Case 3: update interval = 60 seconds



```
int main()
{
    //here, 4 traffic light objects and one i
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    Intersection I1;
    //the traffic light objects were added to
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I1.addLight(L4);
    //the simulation for the intersection sta
    //the yellow timing and update interval w
    //simulation and passed to the run() func
    const int yellowTiming = 3;
    const long long int updateInterval = 60;

    I1.run(yellowTiming, updateInterval);

    return 0;
}
```



```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 60 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 40
Green timing for a traffic light with an ID 1 is: 10
Green timing for a traffic light with an ID 2 is: 8
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 11

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
waiting... 2
```

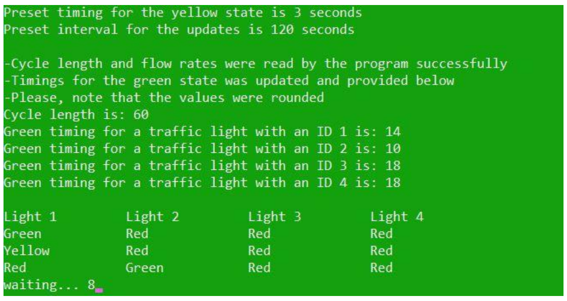


```
Green timing for a traffic light with an ID 1 is: 10
Green timing for a traffic light with an ID 2 is: 8
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 11

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red
Red             Green           Red             Red
Red             Yellow          Red             Red
Red             Red             Green           Red
Red             Red             Yellow          Red
Red             Red             Red             Green
Red             Red             Red             Yellow
Green           Red             Red             Red

-60 seconds have passed since the last update

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 25
Green timing for a traffic light with an ID 1 is: 6
Green timing for a traffic light with an ID 2 is: 3
Green timing for a traffic light with an ID 3 is: 9
Green timing for a traffic light with an ID 4 is: 6

Light 1         Light 2         Light 3         Light 4
Yellow          Red             Red             Red
waiting...
```

When 60 seconds passed, the data was updated. The state was displayed with old parameters, and only after that it displayed the update message and then it followed the new parameters

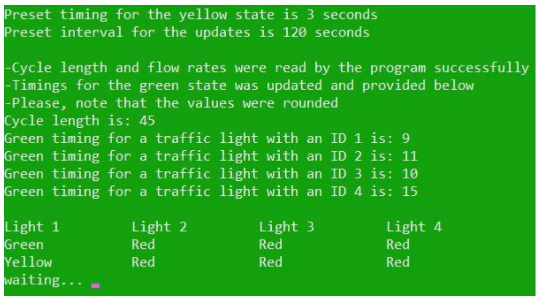
6. For the Simulation Tests

Test Case 1: double values



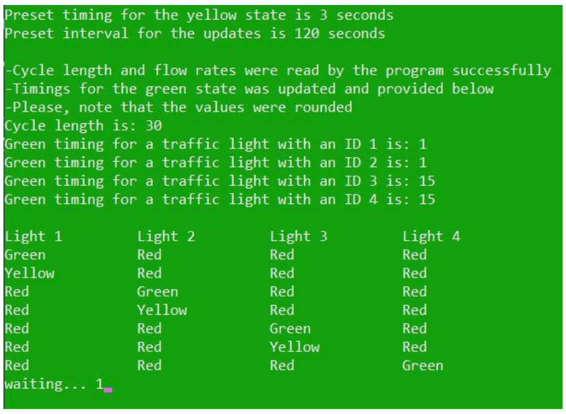
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 60
Green timing for a traffic light with an ID 1 is: 14
Green timing for a traffic light with an ID 2 is: 10
Green timing for a traffic light with an ID 3 is: 18
Green timing for a traffic light with an ID 4 is: 18

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red
Red             Green           Red             Red
waiting... 8
```

Test Case 2: integer values



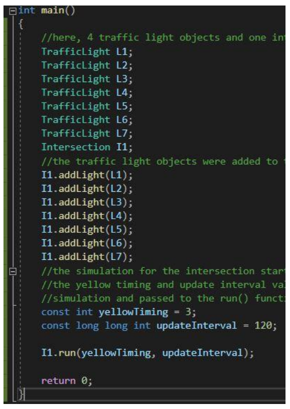
```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 45
Green timing for a traffic light with an ID 1 is: 9
Green timing for a traffic light with an ID 2 is: 11
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 15

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red
waiting...
```

Test Case 3: extreme values

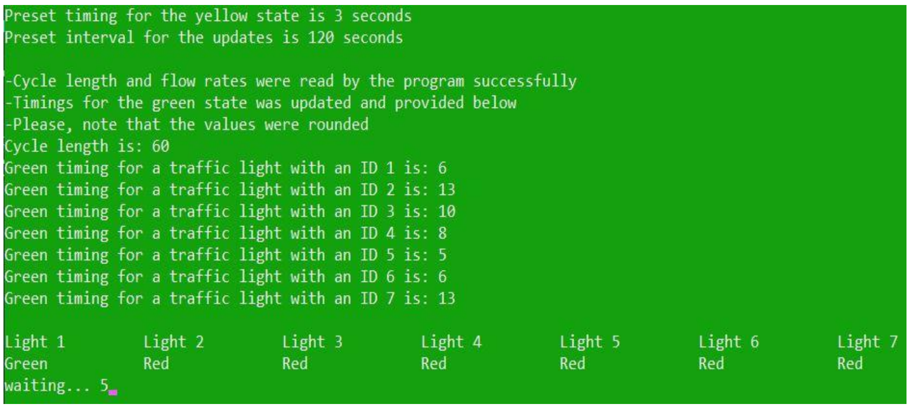


```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 30
Green timing for a traffic light with an ID 1 is: 1
Green timing for a traffic light with an ID 2 is: 1
Green timing for a traffic light with an ID 3 is: 15
Green timing for a traffic light with an ID 4 is: 15

Light 1         Light 2         Light 3         Light 4
Green           Red             Red             Red
Yellow          Red             Red             Red
Red             Green           Red             Red
Red             Yellow          Red             Red
Red             Red             Green           Red
Red             Red             Yellow          Red
Red             Red             Red             Green
waiting... 1
```

Test Case 4:more than 4 traffic lights

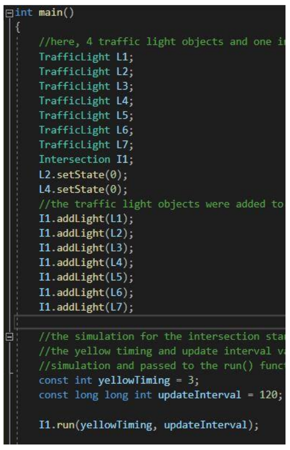


```
int main()
{
    //here, 4 traffic light objects and one in
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    TrafficLight L6;
    TrafficLight L7;
    Intersection I1;
    //the traffic light objects were added to t
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I1.addLight(L4);
    I1.addLight(L5);
    I1.addLight(L6);
    I1.addLight(L7);
    //the simulation for the intersection start
    //the yellow timing and update interval va
    //simulation and passed to the run() functi
    const int yellowTiming = 3;
    const long long int updateInterval = 120;

    I1.run(yellowTiming, updateInterval);

    return 0;
}
```

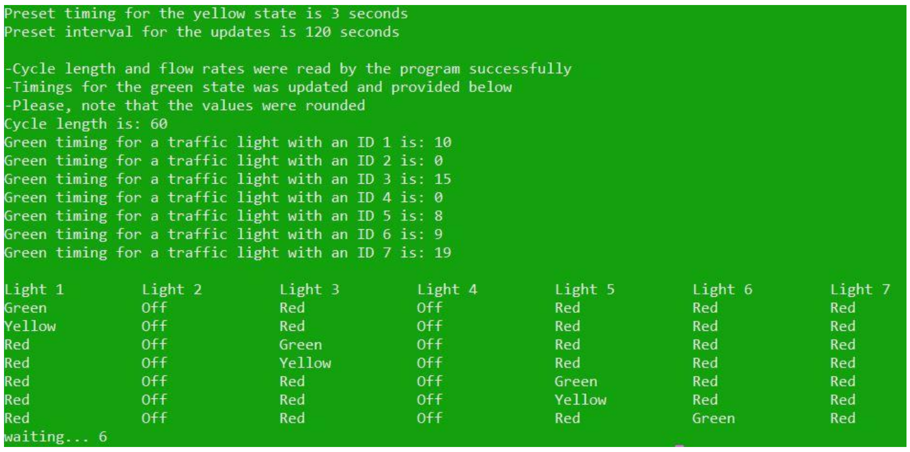


```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 60
Green timing for a traffic light with an ID 1 is: 6
Green timing for a traffic light with an ID 2 is: 13
Green timing for a traffic light with an ID 3 is: 10
Green timing for a traffic light with an ID 4 is: 8
Green timing for a traffic light with an ID 5 is: 5
Green timing for a traffic light with an ID 6 is: 6
Green timing for a traffic light with an ID 7 is: 13

Light 1          Light 2          Light 3          Light 4          Light 5          Light 6          Light 7
Green            Red              Red              Red              Red              Red              Red
waiting... 5
```

Test Case 5: some of the lights are turned off

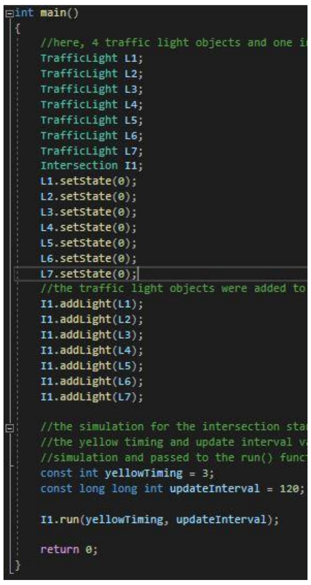


```
int main()
{
    //here, 4 traffic light objects and one i
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    TrafficLight L6;
    TrafficLight L7;
    Intersection I1;
    L2.setState(0);
    L4.setState(0);
    //the traffic light objects were added to
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I1.addLight(L4);
    I1.addLight(L5);
    I1.addLight(L6);
    I1.addLight(L7);

    //the simulation for the intersection sta
    //the yellow timing and update interval v
    //simulation and passed to the run() func
    const int yellowTiming = 3;
    const long long int updateInterval = 120;

    I1.run(yellowTiming, updateInterval);
```

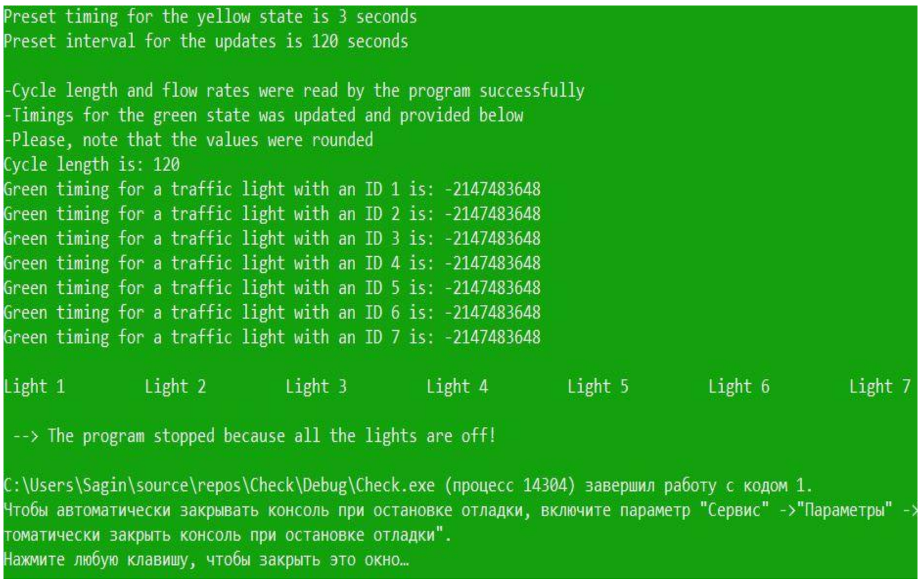


```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 60
Green timing for a traffic light with an ID 1 is: 10
Green timing for a traffic light with an ID 2 is: 0
Green timing for a traffic light with an ID 3 is: 15
Green timing for a traffic light with an ID 4 is: 0
Green timing for a traffic light with an ID 5 is: 8
Green timing for a traffic light with an ID 6 is: 9
Green timing for a traffic light with an ID 7 is: 19

Light 1          Light 2          Light 3          Light 4          Light 5          Light 6          Light 7
Green            Off              Red              Off              Red              Red              Red
Yellow           Off              Red              Off              Red              Red              Red
Red              Off              Green            Off              Red              Red              Red
Red              Off              Yellow           Off              Red              Red              Red
Red              Off              Red              Off              Green            Red              Red
Red              Off              Red              Off              Yellow           Red              Red
Red              Off              Red              Off              Red              Green            Red
waiting... 6
```

Test Case 6: all the lights are turned off



```
int main()
{
    //here, 4 traffic light objects and one i
    TrafficLight L1;
    TrafficLight L2;
    TrafficLight L3;
    TrafficLight L4;
    TrafficLight L5;
    TrafficLight L6;
    TrafficLight L7;
    Intersection I1;
    L1.setState(0);
    L2.setState(0);
    L3.setState(0);
    L4.setState(0);
    L5.setState(0);
    L6.setState(0);
    L7.setState(0);
    //the traffic light objects were added to
    I1.addLight(L1);
    I1.addLight(L2);
    I1.addLight(L3);
    I1.addLight(L4);
    I1.addLight(L5);
    I1.addLight(L6);
    I1.addLight(L7);

    //the simulation for the intersection sta
    //the yellow timing and update interval v
    //simulation and passed to the run() func
    const int yellowTiming = 3;
    const long long int updateInterval = 120;

    I1.run(yellowTiming, updateInterval);

    return 0;
}
```



```
Preset timing for the yellow state is 3 seconds
Preset interval for the updates is 120 seconds

-Cycle length and flow rates were read by the program successfully
-Timings for the green state was updated and provided below
-Please, note that the values were rounded
Cycle length is: 120
Green timing for a traffic light with an ID 1 is: -2147483648
Green timing for a traffic light with an ID 2 is: -2147483648
Green timing for a traffic light with an ID 3 is: -2147483648
Green timing for a traffic light with an ID 4 is: -2147483648
Green timing for a traffic light with an ID 5 is: -2147483648
Green timing for a traffic light with an ID 6 is: -2147483648
Green timing for a traffic light with an ID 7 is: -2147483648

Light 1          Light 2          Light 3          Light 4          Light 5          Light 6          Light 7

 --> The program stopped because all the lights are off!

C:\Users\Sagin\source\repos\Check\Debug\Check.exe (процесс 14304) завершил работу с кодом 1.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно…
```

which are in agreement with the test cases expected output.

Thus, we conclude that the program is functioning correctly since all test cases are verified.

Important Notes:

1. *To execute the program, the user needs to compile and run the code found in the file named Source.cpp*
2. *There is no menu to interact with a user. When the program is run, it starts the simulation automatically and no actions are required from the user.*
3. *The data (flow rates and cycle length) should be included in a txt file called SensorData.txt and be written in separate lines. The SensorData.txt is assumed to be located in the project folder together with Source.cpp*
4. *It is assumed that cycle length is a positive integer, while flow rates are non-negative real numbers.*
5. *The green timings are calculated according to the provided formula in the step 2.*
6. *The cycle length in the data file is assumed to be the summation of green timings only.*
7. *The simulation runs while at least one of the lights is working. If all the lights are turned off at some point the program is terminated.*
8. *The green timings of the turned off lights are set to 0, and the cycle length time is distributed to other lights proportionally among the turned-on lights only.*
9. *The green timings are rounded to the nearest integer and green timings cannot be 0 after the simulation is started. If the calculations lead to it being 0, then it is set to 1.*
10. *When the update interval time passes, the program waits for the occurring state timing and only after that updates the data members.*