Problem Identification and Statement

Design a software to control a system of traffic lights at an intersection. The software should simulate the behavior of traffic lights at an intersection.

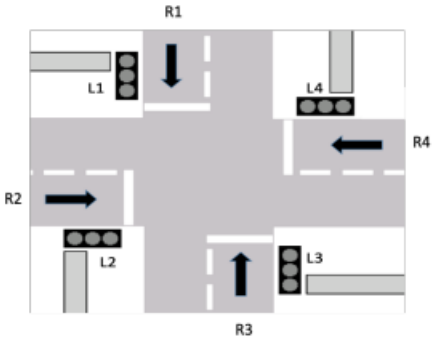
Cycle length and traffic flow information must be read from a file. Every specific duration (say 24 hours), the data must be read from the file, the green timings should be updated based on the latest traffic condition, and the control should proceed with the updated green timings.

Gathering Information

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. The sensors save the traffic rate information into a file. The system is represented in the picture below:



3. the four traffic lights are represented as L1, L2, L3, and L4. The system operates as follows:

- 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.

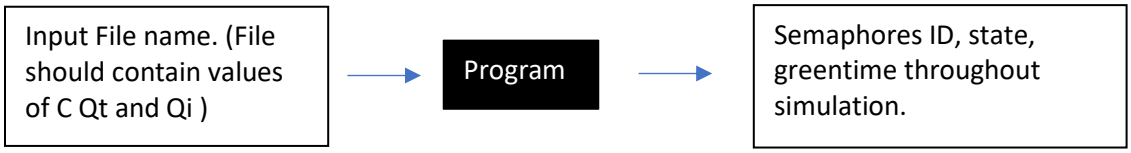
- i. The next cycle starts with traffic light L1 becoming green again, and so on.

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, and C represents the cycle length in seconds.

I/O diagram



Step 3: Test Cases and algorithm

Test Cases:

The text document Traffic_Information was provided to our program. It contains Cycle Length, Total Traffic Flow, and Traffic Flow values for each trafficlight.

The document Traffic_Information was updated as shown in figure 2 after 60 seconds to test the function updateTiming()

Figure 1: Traffic information dataset 1

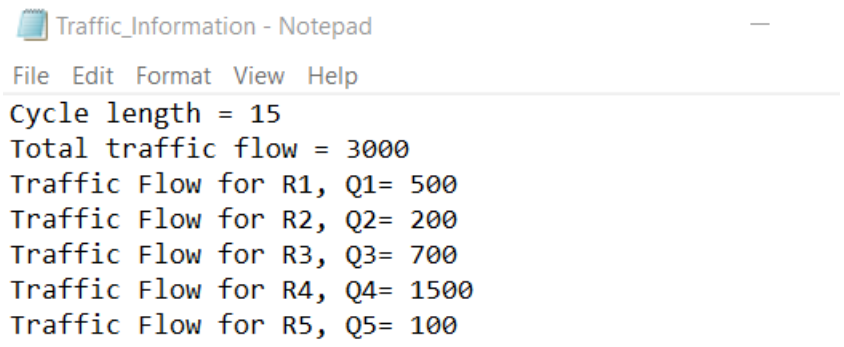
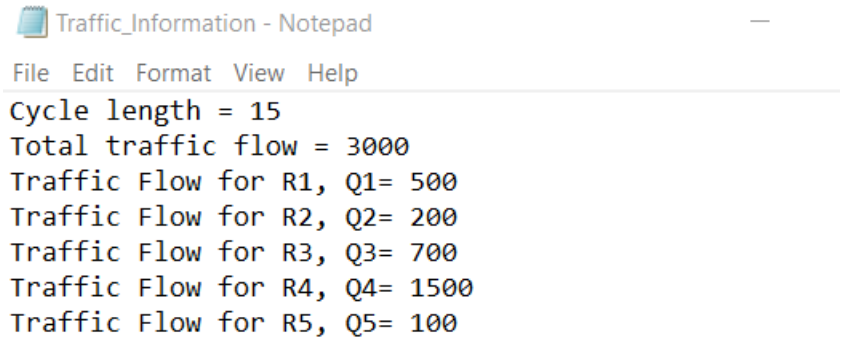


Figure 2: Traffic Information dataset 2



The value of green time for the first semaphore given dataset 1 can be obtained as follows

$$d1 = \frac{q1}{qt} C = \frac{500}{500 + 200 + 700 + 1500 + 100} = 2.5$$

Similarly, green time values for each semaphore can be calculated. Results are summarized in the table below.

Table 1:

SEMAPHORE NUMBER	TRAFFIC FLOW Q	GREEN TIME D	CONSTANTS
1	500	2.5	Cycle Length C
2	200	1	15
3	700	3.5	
4	1500	7.5	Total Traffic Flow Qt
5	100	0.5	3000
1	800	5	Cycle Length C
2	200	1.25	20
3	100	0.625	
4	1700	10.625	Total Traffic Flow Qt
5	400	2.5	3200

Expected time for one cycle of simulation given dataset 1 can be found as:

$$t_1 = d_1 + d_2 + d_3 + d_4 + d_5 + 5y_t = 2.5 + 1 + 3.5 + 7.5 + 0.5 + 3 \times 5 = 30s$$

Where y_t is time semaphore remains yellow: $y_t = 3s$

Similarly expected time for one cycle of simulation given dataset 2 can be found as:

$$t_1 = d_1 + d_2 + d_3 + d_4 + d_5 + 5y_t = 5 + 1.25 + 0.625 + 10.625 + 2.5 + 3 \times 5 = 35s$$

Therefore,

To test run() the state and ID of semaphores are displayed on console three times a cycle for each semaphore. Values of state of semaphores during simulation should change from 3 to 2 to 1, ID values of semaphores should not change during the simulation. The simulation should not end (unless the user exits the program) to properly represent real life situation.

To test readTrafficData(), printLightInfo(), CalculateGreenTime(), UpdateTiming() values of C, Traffic Flow values, green time values will be printed on the console. These values must correspond to values in table 1.

To test AddLight() and getNumOfSemaphores(), 5 semaphores will be added to our Intersection and the number of Semaphores NumOfSemaphores will be printed. We expect NumOfSemaphores to be 5. To test droplight() we will remove one semaphore, we expect NumOfSemaphore to decrease by 1 from 5 to 4.

To test wait() we will make use of a timer. We will time each cycle of the simulation. The time of one cycle of simulation given dataset 1 is expected to be 30s; The time of one cycle of simulation given dataset 2 is expected to be 35s. Values of green time for each semaphore should be updated after 60 seconds

Pseudocode

```
Define class TrafficLights
	Private members
		ID as integer
		State as integer
		Greentime as double
		TLnumber as integer, global variable

	Public members
		TrafficLights()
			Assign TLnumber to ID
			Increment TLnumber
			Assign 1 to state
			Assign 0 to greentime
		getID()
			return ID
		getState()
			return state
		setState(color)
			Assign color to state
		getGreentime()
			return greentime
		setGreentime(GreenTime)
			Assign GreenTime to greentime
		PrintLightInfo()
			Print "ID = ", ID, newline
			Print "state = ", state, newline
			Print "greentime = ", greentime, newline
		getNumOfTrafficlights()
			return TLnumber
		wait(number_seconds)
			Assign clock() to startClock
			Assign number_seconds*CLOCKS_PER_SEC to SecondsWaited
```

```
                    while clock() less than startClock + SecondsWaited
                           return

Assign 0 to TLnumber
Assign 3 to yellowtime
Assign 60 to updategreentime
Assign 10 to tMAX


Define class Intersection
       Private members
              Define trafficlights as array of TrafficLights, and maximum size tMAX
              Define CycleLength as double
              Define TotalTrafficFlow as double
              Define TrafficFlows as array of double and maximum size tMAX
              Define GreenTime as array of double and maximum size tMAX
              Define TLnumber as Integer

       Public members
              Intersection()
                           Assign 0 to TLnumber
                           Assign 0 to CycleLength
                           Assign 0 to TotalTrafficFlow
                           For i=0, until i smaller than tMAX, increment i at each
                           iteration
                                    Assign 0 to TrafficFlows[i]
                                    Assign 0 to GreenTime[i]
              AddLight(aLight as TrafficLights) {
                     if TLnumber less than tMAX)
                           assign aLight to trafficlights[TLnumber]
                           increment TLnumber
                     otherwise
                           Print "Can't add more traffic lights, full capacity", newline
              dropLight(TLightID as integer) {
                     Assign false to isFound
                     For i=0, until i smaller than tMAX, increment i at each iteration
                                    if TLightID equals return value when trafficlights[i]
                                    calls getID()
                                           Print "The trafficlight to be removed is found",
                                           newline
                                           for j = i; j smaller TLnumber; increment j at
                                           each iteration
                                                  Assign trafficlights[j + 1] to
                                                  trafficlights[j]
                                           Decrement TLnumber
                                           Assign true to isFound


                     if isFound is negated
                           Print "Can't find the traffic light to drop" , newline

              getNumOfSemaphores()
                     return TLnumber;

              ReadTrafficData()
                     Define infile as buffer
                     Infile opens "Traffic_Information.txt" as input file
                     if infile does not open file correctly
```

```
                Print "error in opening file"
                Terminate Program and return (-1)
        Define trash as string
        Store words from infile, into trash, all the words until sign =
        Assign input value from infile to CycleLength
        Print "CycleLength = ", CycleLength, newline
        Store words from infile, into trash, all the words until sign =
        Assign input value from infile to TotalTrafficFlow
        Print "TotalTrafficFlow = ",TotalTrafficFlow, newline

        for i = 0, i less than TLnumber, increment i at each iteration
                if infile reaches the end of file
                        exit loop
                Store words from infile, into trash, all the words until sign
                =
                Assign input value from infile to TrafficFlows[i];
                Print "traffic Flow for semaphore", I, " = ", TrafficFlows[i],
                newline
        Infile closes input text file

CalculateGreenTime()
        for i = 0, i less than TLnumber, increment i at each iteration
                Assign (TrafficFlows[i]) * (CycleLength / TotalTrafficFlow)
                to greentime
                Assign greentime to GreenTime[i]
                Print "Computed Green Time for semaphore ", i, " = "
                greentime," s", newline
                trafficlights[i] calls setGreentime(GreenTime[i]);
                Print "Updated Green Time for semaphore ",i," is now set to =
                ", trafficlights[i] calls getGreentime(), newline
                Print "Updated information of semaphore ", I," as follows:
                “,newline
                trafficlights[i] calls PrintLightInfo();
                Print newline
run()
Assign 0 to TimeSimulation
Print "Simulation starts", newline, newline
for 1 equals 1
        Assign 0 to start
        Assign 0 to endCycle
        Assign 0 to TimeOneCycle
        Assign clock() to start
        for i = 0, i less than TLnumber, increment i at each iteration
                Print "Cycle of Semaphore " i, newline, newline
                trafficlights[i] calls setState(1);
                trafficlights[i] calls setState(3);
                 if return value when trafficlights[i] calls getState() does
                not equal 3
                        Continue to next iteration of loop
                Otherwise
                        trafficlights[i] calls PrintLightInfo();
                        trafficlights[i] calls wait(result of trafficlights[i]
                calls getGreentime());
                        trafficlights[i] calls setState(2);
                        trafficlights[i] calls PrintLightInfo();
                        trafficlights[i] calls wait(yellowtime);
                        trafficlights[i] calls setState(1);
                        trafficlights[i] calls PrintLightInfo();
```

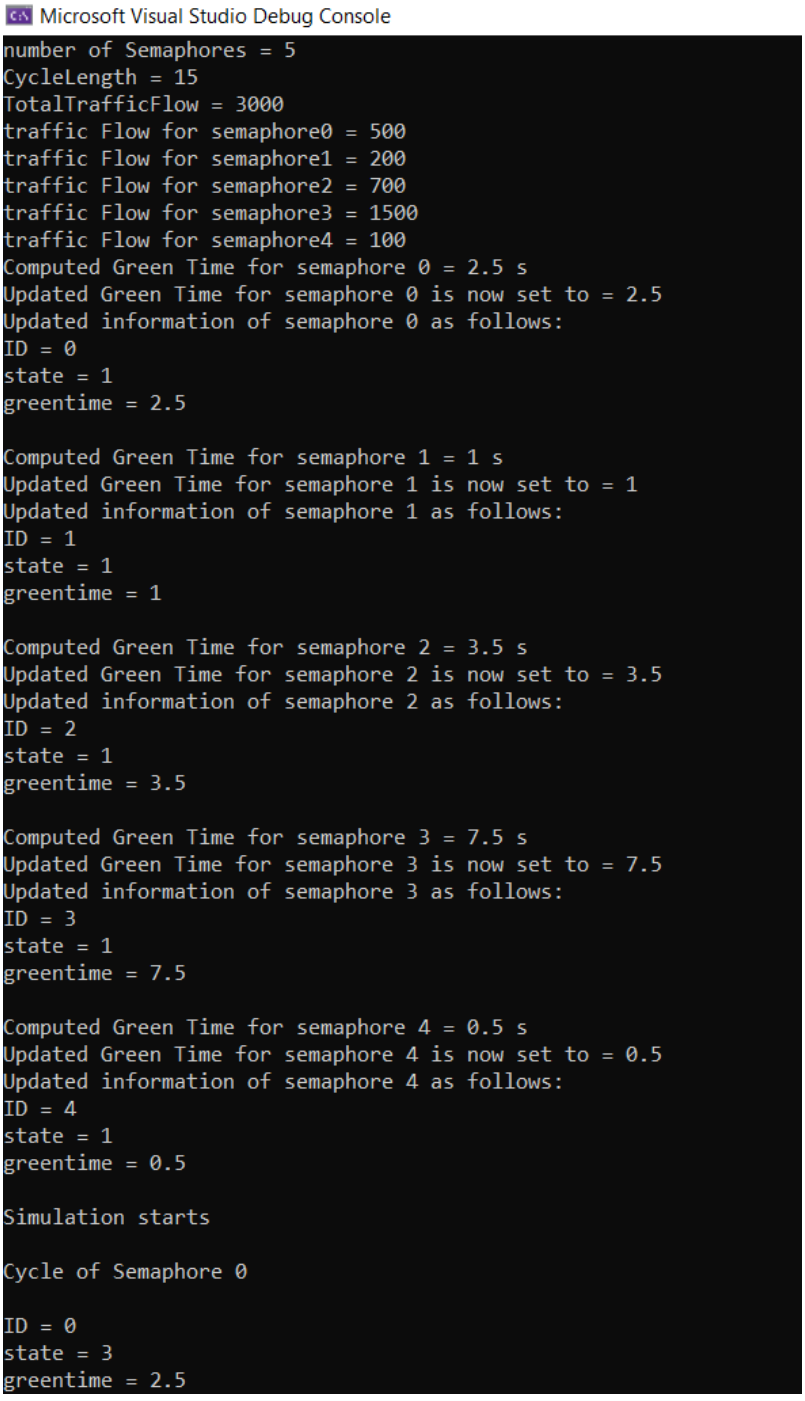
```
                        Print newline
                Assign clock() to endCycle
                Assign endCycle – start to TimeOneCycle
                Assign TimeSimulation + TimeOneCycle to TimeSimulation
                if TimeSimulation greater than updategreentime x CLOCKS_PER_SEC
                        call UpdateTiming();
                        Assign 0 to TimeSimulation = 0;


        void UpdateTiming()
                call ReadTrafficData()
                call CalculateGreenTime();


Main function
main() returns an integer value
        Define Semaphore1, Semaphore2, Semaphore3, Semaphore4, Semaphore5 as TrafficLights
        Define WashingtonRoad3 as Intersection
        WashingtonRoad3 calls AddLight(), giving Semaphore1 as parameter
        WashingtonRoad3 calls AddLight(), giving Semaphore2 as parameter
        WashingtonRoad3 calls AddLight(), giving Semaphore3 as parameter
        WashingtonRoad3 calls AddLight(), giving Semaphore4 as parameter
        WashingtonRoad3 calls AddLight(), giving Semaphore5 as parameter
        Print "number of Semaphores = ", WashingtonRoad3 calls getNumOfSemaphores(),
        newline
        WashingtonRoad3 calls ReadTrafficData();
        WashingtonRoad3 calls CalculateGreenTime();
        for 1 equals 1 then
                WashingtonRoad3 calls run()
        return 0
```

Code or implementation

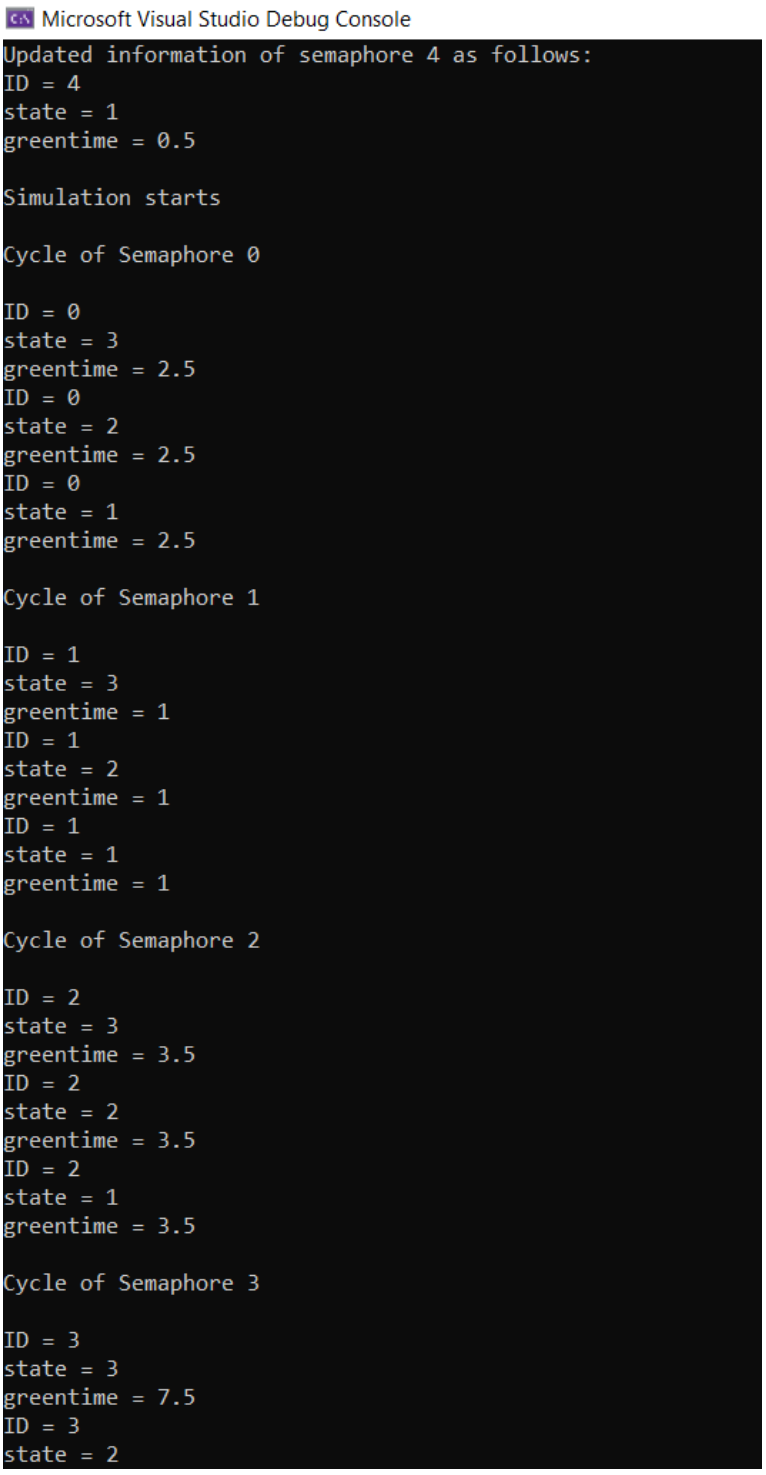
The code is presented in the documents attached: Trotolo_Assignment4.cpp, TrafficLight.h, Intersection.h

Test and Verification

Figure 1: readTrafficData(); printLightInfo(); `CalculateGreenTime();` are tested. Since values of ID,state, Cycle Length, Traffic flow values, green time values correspond to values in table 1 the functions are working correctly. AddLight() and getNumOfSemaphores() are also working since number of semaphore in intersection is 5, and public functions of TrafficLight class are working (which implies semaphores are correctly added to the intersection)



Figures 2 to 5: run() function is tested. Notice that values of state for each semaphore vary as predicted



Microsoft Visual Studio Debug Console

```
greentime = 3.5
ID = 2
state = 1
greentime = 3.5

Cycle of Semaphore 3

ID = 3
state = 3
greentime = 7.5
ID = 3
state = 2
greentime = 7.5
ID = 3
state = 1
greentime = 7.5

Cycle of Semaphore 4

ID = 4
state = 3
greentime = 0.5
ID = 4
state = 2
greentime = 0.5
ID = 4
state = 1
greentime = 0.5

Cycle of Semaphore 0

ID = 0
state = 3
greentime = 2.5
ID = 0
state = 2
greentime = 2.5
ID = 0
state = 1
greentime = 2.5

Cycle of Semaphore 1

ID = 1
state = 3
greentime = 1
ID = 1
state = 2
greentime = 1
ID = 1
```

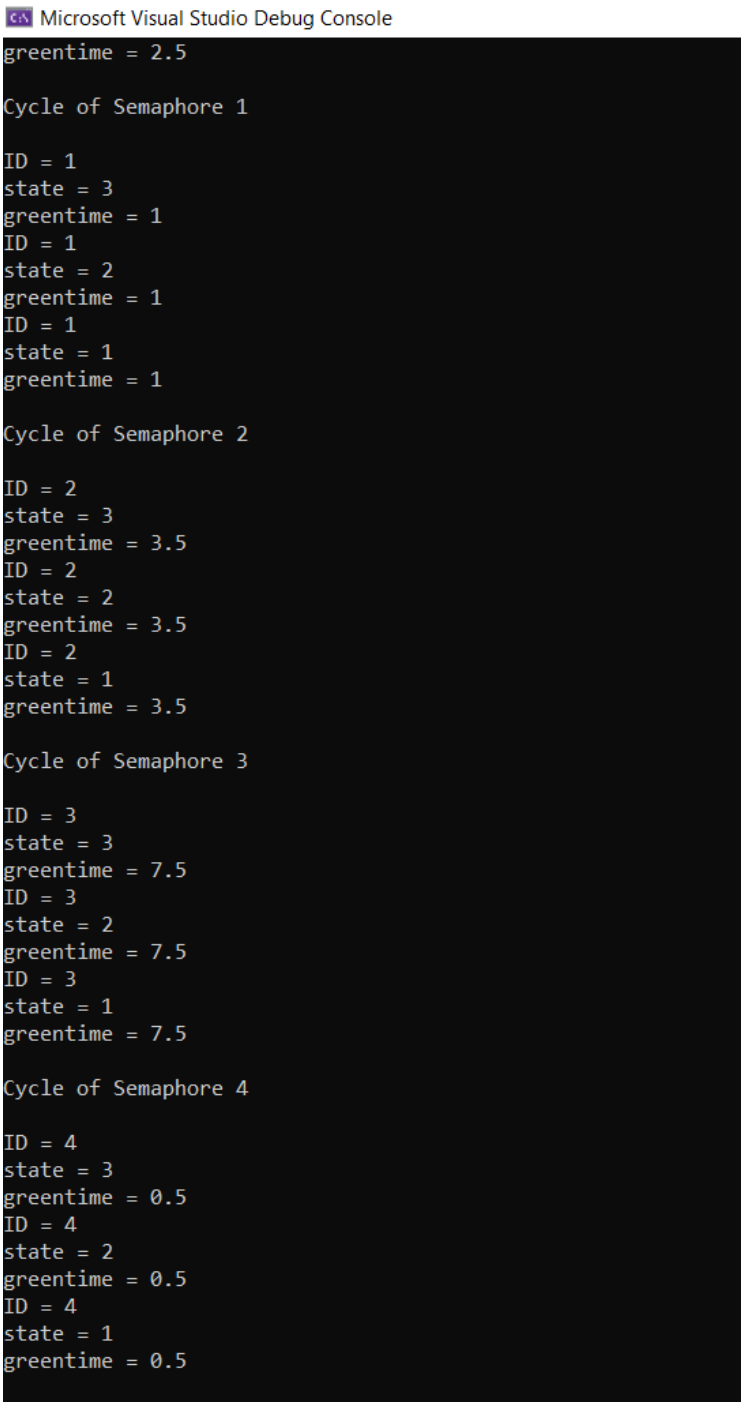
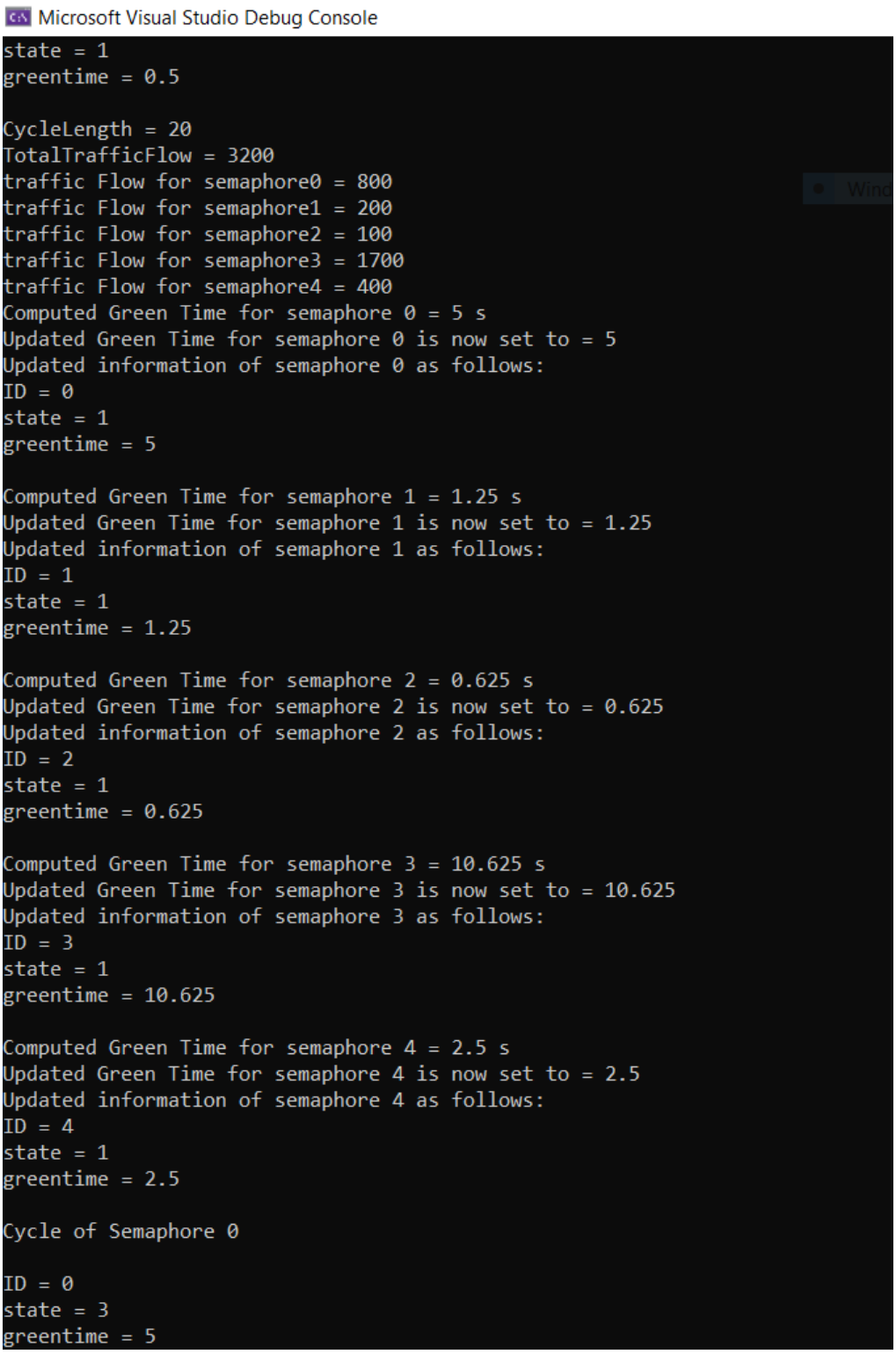
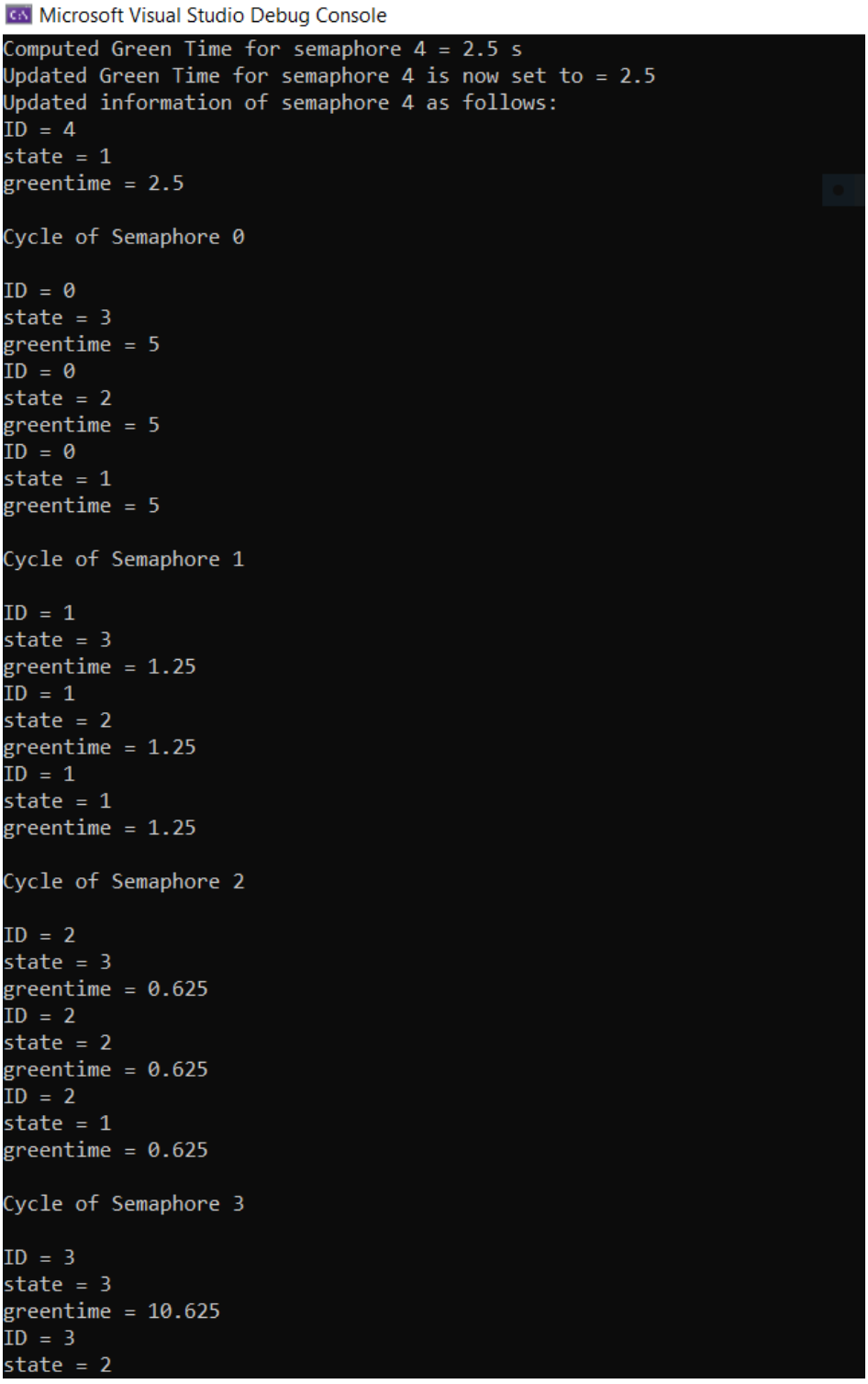

Microsoft Visual Studio Debug Console
greentime = 2.5
Cycle of Semaphore 1
ID = 1
state = 3
greentime = 1
ID = 1
state = 2
greentime = 1
ID = 1
state = 1
greentime = 1
Cycle of Semaphore 2
ID = 2
state = 3
greentime = 3.5
ID = 2
state = 2
greentime = 3.5
ID = 2
state = 1
greentime = 3.5
Cycle of Semaphore 3
ID = 3
state = 3
greentime = 7.5
ID = 3
state = 2
greentime = 7.5
ID = 3
state = 1
greentime = 7.5
Cycle of Semaphore 4
ID = 4
state = 3
greentime = 0.5
ID = 4
state = 2
greentime = 0.5
ID = 4
state = 1
greentime = 0.5

Figure 6: `UpdateTiming() is tested.` Values of green time of semaphores are updated.



Figures 7 to 10: run () function tested after `UpdateTiming()` control proceeds with the updated green timings without abnormality



Microsoft Visual Studio Debug Console

```
ID = 2
state = 1
greentime = 0.625

Cycle of Semaphore 3

ID = 3
state = 3
greentime = 10.625
ID = 3
state = 2
greentime = 10.625
ID = 3
state = 1
greentime = 10.625

Cycle of Semaphore 4

ID = 4
state = 3
greentime = 2.5
ID = 4
state = 2
greentime = 2.5
ID = 4
state = 1
greentime = 2.5

Cycle of Semaphore 0

ID = 0
state = 3
greentime = 5
ID = 0
state = 2
greentime = 5
ID = 0
state = 1
greentime = 5

Cycle of Semaphore 1

ID = 1
state = 3
greentime = 1.25
ID = 1
state = 2
greentime = 1.25
ID = 1
state = 1
```

Microsoft Visual Studio Debug Console

```
greentime = 5

Cycle of Semaphore 1

ID = 1
state = 3
greentime = 1.25
ID = 1
state = 2
greentime = 1.25
ID = 1
state = 1
greentime = 1.25

Cycle of Semaphore 2

ID = 2
state = 3
greentime = 0.625
ID = 2
state = 2
greentime = 0.625
ID = 2
state = 1
greentime = 0.625

Cycle of Semaphore 3

ID = 3
state = 3
greentime = 10.625
ID = 3
state = 2
greentime = 10.625
ID = 3
state = 1
greentime = 10.625

Cycle of Semaphore 4

ID = 4
state = 3
greentime = 2.5
ID = 4
state = 2
greentime = 2.5
ID = 4
state = 1
greentime = 2.5
```

The simulation does never terminate, unless the debugged is stopped by the user. This is what we decided.

Id, State and Green Time of each semaphore are consistent with the predicted values. Time consistency is also respected, which is shown with the help of a timer:

01 is the first cycle of the simulation (30s as expected)

02 is the second cycle of the simulation (30s as expected)

(Green time values updated after 60 s; This corresponds to figure 6.)

03 is the third cycle of the simulation (35s as expected)

04 is the fourth cycle of the simulation (35s as expected)

Hence wait(), `UpdateTiming()` and `run()` are working correctly

Microsoft Visual Studio Debug Console

```
number of Semaphores = 5
CycleLength = 20
TotalTrafficFlow = 3200
traffic Flow for semaphore0 = 800
traffic Flow for semaphore1 = 200
traffic Flow for semaphore2 = 100
traffic Flow for semaphore3 = 1700
traffic Flow for semaphore4 = 400
Computed Green Time for semaphore 0 = 5 s
Updated Green Time for semaphore 0 is now set to = 5
Updated information of semaphore 0 as follows:
ID = 0
state = 1
greentime = 5

Computed Green Time for semaphore 1 = 1.25 s
Updated Green Time for semaphore 1 is now set to = 1.25
Updated information of semaphore 1 as follows:
ID = 1
state = 1
greentime = 1.25

Computed Green Time for semaphore 2 = 0.625 s
Updated Green Time for semaphore 2 is now set to = 0.625
Updated information of semaphore 2 as follows:
ID = 2
state = 1
greentime = 0.625

Computed Green Time for semaphore 3 = 10.625 s
Updated Green Time for semaphore 3 is now set to = 10.625
Updated information of semaphore 3 as follows:
ID = 3
state = 1
greentime = 10.625

Computed Green Time for semaphore 4 = 2.5 s
Updated Green Time for semaphore 4 is now set to = 2.5
Updated information of semaphore 4 as follows:
ID = 4
state = 1
greentime = 2.5

The trafficlight to be removed is found
number of Semaphores = 4

C:\Users\user\Desktop\ass4\Debug\ass4.exe (process 11080) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debu
Press any key to close this window . . .
```

Figure 12: droplight() is working properly since the number of traffic lights is reduced from 5 to 4