Name: Muhammad Zaeem Shahzad
Student ID: ms12297
University ID: N16933069
Date: November 28, 2020

Assignment 4 Report: Traffic Light Control System

Step 1 - Problem Identification and Statement:

The program simulates the behavior of a number of traffic lights at an intersection based on information read from a data file (where sensory data is written into). 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.

Step 2 - Gathering Information:

- *Background Information:*

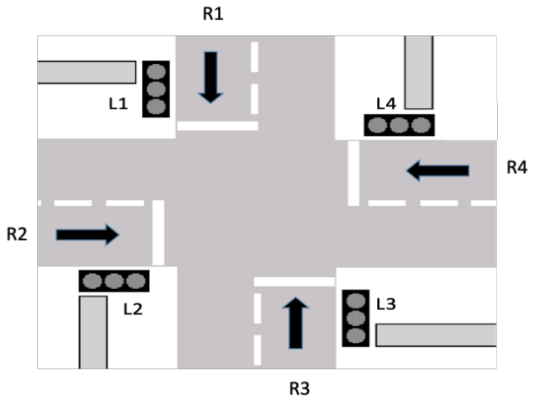


Figure 1: Intersection with traffic lights (L stands for traffic light while R stands for road).

The system that the program simulates has the following components:

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

3) *The signals operate in a conventional fashion. Traffic is allowed to move on one road, say R1, and then the next (R2), alternatively across the four roads of the intersection. Assume that the four traffic lights are represented as L1, L2, L3, and L4. The system operates as follows:*
   a) *Traffic light (L1) is green for a duration calculated based on the traffic flow rate in road R1, the other traffic lights (L2, L3, and L4) are red.*
   b) *L1 becomes yellow for X seconds (X being a constant value). The Department of Transportation's traffic manual recommends that yellow lights are between 3 and 6 seconds long. Other traffic lights (L2, L3, and L4) remain in red state.*
   c) *Then, traffic light L2 becomes green for a duration calculated based on the traffic flow rate in road R2. Meanwhile, L1, L3, and L4 are red.*
   d) *Traffic light L2 becomes yellow for X seconds (X being a constant value). Other traffic lights (L1, L3, and L4) remain in red state.*
   e) *Then, traffic light L3 becomes green for a duration calculated based on the traffic flow rate in road R3. Meanwhile, traffic lights L1, L2, and L4 are red.*
   f) *Traffic light L3 becomes yellow for X seconds (X being a constant value). Other traffic lights (L1, L2, and L4) remain in red state.*
   g) *Then, traffic light L4 becomes green for a duration calculated based on the traffic flow rate in road R4. Meanwhile, traffic lights L1, L2, and L3 are red.*
   h) *Traffic light L4 becomes yellow for X seconds (X being a constant value). Other traffic lights (L1, L2, and L3) remain in red state.*
   i) *The next cycle starts with traffic light L1 becoming green again, and so on.*

4) *The green timings for the traffic lights are updated regularly based on traffic flow. 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.*

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 i-th traffic light, Q_i represents the traffic flow (number of vehicles per hour) crossing the i-th traffic light, Q_T represents the total traffic flow passing through the intersection, and C represents the cycle length in seconds. 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). 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.

- *Inputs/Outputs:*
  - *The inputs are:*
    - *Number of traffic lights at intersection*
    - *Name of data file that contains traffic information*
    - *Input of 1 or 0 in the Exit Display is prompted if the user inputs a number less than or equal to 0 as the number of traffic lights at intersection*
  - *The outputs are:*
    - *The simulation starts*
    - *The computed green timings are displayed*
    - *The change in state of traffic lights is displayed*
    - *The total time of simulation is displayed after every cycle*
    - *After an interval of 1.5 minutes, green timings are updated and the updated timings are displayed.*
    - *The simulation starts anew with the updated green timings.*
  - *For invalid inputs:*
    - *The program prompts for correct input again after displaying an error message. Only valid input is accepted.*

Step 3 - Test Cases and Algorithm:

1) *Test Cases:*

- *Test Case 1 – Using 4 traffic lights and input file case1.txt:*

Contents of input file case1.txt:
C = 30
Q1 = 800
Q2 = 750
Q3 = 950
Q4 = 2000

For input of 4 in traffic light number and case1.txt in file name, the program calculates the following respective green-times:

Total green-time of all traffic lights = $C - (yellowtime \times no.of\ traffic\ lights)$
= 30 – (3 x 4) = 18

Traffic Light 1 = 3.2 seconds
Traffic Light 2 = 3 seconds
Traffic Light 3 = 3.8 seconds
Traffic Light 4 = 8 seconds

And the time of simulation after one cycle should be approximately 30 seconds.

An update is called after the time of simulation becomes equal to or greater than 90 seconds. The same simulation continues from the start because the contents of case1.txt have not been changed.

- *Test Case 2 – Using 5 traffic lights and input file case2.txt:*

Contents of input file case2.txt:
C = 55
Q1 = 1000
Q2 = 800
Q3 = 1200
Q4 = 1500
Q5 = 500

For input of 5 in traffic light number and case2.txt in file name, the program calculates the following respective green-times:

Total green-time of all traffic lights = $C - (yellowtime \times no.of\ traffic\ lights)$
= 55 – (3 x 5) = 40

Traffic Light 1 = 8 seconds
Traffic Light 2 = 6.4 seconds
Traffic Light 3 = 9.6 seconds
Traffic Light 4 = 12 seconds
Traffic Light 5 = 4 seconds

And the time of simulation after one cycle should be approximately 55 seconds.

An update is called after the time of simulation becomes equal to or greater than 90 seconds. The same simulation continues from the start because the contents of case2.txt have not been changed.

- *Test Case 3 – Input Validation:*

The program validates all inputs, namely inputs for traffic light number, the name of the data file, and the choice in the exit display. A set of invalid inputs are tested in this case. In the case of file name and traffic light number, the program displays an error message and prompts for input again. In case of invalid input in the exit display, the program directs the user to the start of the program.

- *Test Case 4 – Termination of Program after input of 0 in traffic light number:*

Upon input of 0 for traffic light number, the program displays the confirmation message, "You chose to exit, please confirm". Then the "Return to Start or Exit" choice is displayed. Upon input of 0, the program terminates.

2) *Algorithm:*

```
Declare a variable that holds a sequence of characters filename
Declare a constant integer variable x, and assign 1 to it
Declare a constant number variable yellowtime and assign 3 to it
Declare a constant number variable updatetime and assign 1.5
multiplied by 60.0 to it
Declare a constant integer variable TLmax and assign 5 to it

Define class TrafficLight:
   Private members:
     Integer variable ID
     Integer variable state
     Number variable greentime
     Integer variable that is independent of class instance presence
     TLnum
   Public members:
     Function TrafficLight:
       Pass into function: Nothing
       Assign TLnum to ID
       Assign 1 to state
       Assign 0 to greentime
       Add 1 to TLnum
       Pass out of function: Nothing
     Function getID:
       Pass into function: Nothing
       Pass out of function: ID
     Function getState:
       Pass into function: Nothing
       Pass out of function: state
     Function getGreentime:
       Pass into function: Nothing
       Pass out of function: greentime
     Function setState:
       Pass into function: integer color
       Assign color to state
       Pass out of function: Nothing
     Function setGreentime:
       Pass into function: number Greentime
       Assign Greentime to greentime
       Pass out of function: Nothing
     Function PrintTLinfo:
       Pass into function: Nothing
```

```
      Print "ID = ", ID
      Print "State = "
      If state is equal to 1, Print "Red"
      If state is equal to 2, Print "Yellow"
      If state is equal to 3, Print "Green"
      If state is equal to 0, Print "Off"
      Print "Greentime = ", greentime
      Leave a line
      Pass out of function: Nothing

    Function independent of presence of class instance get_numofTL:
      Pass into function: Nothing
      Pass out of function: TLnum

    Function wait:
      Pass into function: number seconds
      Declare a clock type variable and assign the processor time
      consumed by the program to it
      Declare a number wait and assign the value of seconds multiplied
      by clock ticks per second to it
      As long as the processor time consumed by the program is less
      than start + wait, repeat:
        No code in loop
      Pass out of function: Nothing

Assign 0 to integer TLnum of class TrafficLight

Define class Intersection:
  Private members:
    An array of TrafficLight instances TL of size TLmax
    Number variable C
    Number variable Q_t
    An array of number variables Q_i of size TLmax
    An array of number variables green_time of size TLmax
    Integer variable TLnum
  Public members:
    Function Intersection:
      Pass into function: Nothing
      Assign 0 to TLnum
      Assign 0 to C
      Assign 0 to Q_t
      As long as integer i, initially 0, is less than TLmax, repeat:
        Assign 0 to Q_i element at index i
        Assign 0 to green_time element at index is
        Add 1 to i
      Pass out of function: Nothing
    Function AddLight:
      Pass into function: Nothing
      If TLnum is less than TLmax:
        Declare a TrafficLight instance a
        Assign a to TL element at index TLnum
        Add 1 to TLnum
```

If TLnum is not less than TLmax, print "Limit reached! Cannot add more traffic lights"
Pass out of function: Nothing

Function droplight:
Pass into function: integer TL_ID
Declare a Boolean variable isTL and assign 0 to it
As long as integer i, initially 0, is less than Tlnum, repeat:
Call the getID function of TL element at index i
If TL_ID is equal to the return value of the previous getID function:
As long as integer j, initially equal to i, is less than TLnum, repeat:
Assign TL element at index j plus 1 to TL element at index j
Add 1 to j
Subtract 1 from TLnum
Print "Removed traffic light ", TL_ID
Assign 1 to isTL
Add 1 to i
If isTL is equal to 0, print "Can't find the traffic light to drop!"
Pass out of function: Nothing

Function get_numofTL:
Pass into function: Nothing
Pass out of function: TLnum

Function readTrafficData:
Pass into function: Nothing
Assign 0 to Q_t
Declare an input file stream variable infile
As long as integer z, initially 1, is equal to 1, repeat:
Open filename in the file stream infile
If the file does not open, print "Error opening file! Please enter the correct file name", and then prompt for input of filename
If the file has opened, end all repetitions
Declare a variable that holds a sequence of characters skip
From infile, input characters into skip till the character '=' is reached
Input into C from infile
Assign C minus the product of yellowtime and TLnum to C
As long as integer i, initially 0, is less than TLnum, repeat:
If the end of file has been reached, end all repetitions
From infile, input characters into skip till the character '=' is reached
Input into Q_i element at index i from infile
Assign the sum of Q_t and Q_i elemenet at index i to Q_t
Add 1 to i
Close the input file of infile
Pass out of function: Nothing

```
Function calculate_greentime:
  Pass into function: Nothing
  As long as integer i, initially 0, is less than TLnum, repeat:
    Assign the product of C and Q_i element at index i divided by
    Q_t to a number variable greentime
    Assign greentime to green_time element at index i
    Call the setGreentime function of TL element at index i with
    input of green_time element at index i
    Print "Green Time for Traffic Light ", i plus 1, " is = ", the
    return value of the function getGreentime of TL element at
    index i, " seconds"
    Add 1 to i
  Pass out of function: Nothing

Function run:
  Pass into function: Nothing
  Declare a number variable SimulationT and assign 0 to it
  Declare an integer variable cycle and assign 1 to it
  Print "Commecning Simulation"
  As long as x is equal to 1, repeat:
    Declare number variable start and assign 0 to it
    Declare number variable end and assign 0 to it
    Declare number variable time_Cycle and assign 0 to it
    Assign the processor time consumed by the program to start
    Print "Cycle ", cycle
    Add 1 to cycle

    As long as integer i, initially 0, is less than TLnum, repeat:
      Call the function setState of TL element at index i with
      input 3
      Print "Traffic Light ", i plus 1, " :"
      Print "Green"
      Call the wait function of TL element at index i with input
      of the return value of function getGreentime of TL element
      at index i
      Call the function setState of TL element at index i with
      input 2
      Print "Yellow"
      Call the wait function of TL element at index i with input
      yellowtime
      Call the function setState of TL element at index i with
      input 1
      Cout "Red"
      Add 1 to i

    Assign the processor time consumed by the program to end
    Assign end minus start to time_Cycle
    Assign the sum of SimulationT and time_Cycle to SimulationT
    Print "Time elapsed since start of simulation: ", SimulationT
    divided by clock ticks per second, " seconds"
```

```
        If SimulationT is greater or equal to updatetime multiplied by
        clock ticks per second:
          Call the updateTiming function
          Print "The time for the updated simulation will now be
          reset"
          Assign 0 to SimulationT
          Assign 1 to cycle
      Pass out of function: Nothing

    Function updateTiming:
      Pass into function: Nothing
      Print "Green times for all traffic lights will now be updated"
      Call the readTrafficData function
      Call the calculate_greentime function
      Pass out of function: Nothing


Main function:
Pass into function: Nothing

Declare an Intersection instance road
Declare an integer TLnum
Declare an integer y and assign 1 to it
Print "Welcome to the Traffic Light Control System Simulator"
Print "You can run a simulation of traffic lights at an intersection
in this program"
Print "Note that the program supports a maximum of 5 traffic lights at
the intersection"

As long as y is equal to 1, repeat:
   Print "Enter the number of traffic lights (not more than 5) you want
   to simulate at the intersection: "
   Print "You may enter 0 to exit program"
   Prompt for input of TLnum
   If the input is not an integer:
     Clear input buffer memory for TLnum
     Print "Please enter a positive integer!"
     Skip this repetition and start a new one
   If the input is an integer:
     If TLnum is greater than 5:
       Print "You must enter an integer value less than 5! Try again"
       Skip this repetition and start a new one
     If TLnum is less than or equal to 0:
       Print "You chose to exit, please confirm"
       Assign 0 to y
     If TLnum is greater than 0 and less than or equal to 5:
       As long as integer i, initially 0, is less than TLnum, repeat:
         Call AddLight function of road
         Add 1 to i
       Print "Enter the name of the file that contains the traffic
       information stored by the sensors: "
       Clear input stream's buffer memory
```

```
        Prompt for input of filename
        Call the readTrafficData function of road
        Call the calculate_greentime function of road
        Call the run function of road

    Print "To"
    Print all the following outputs to the left of the screen
    Print in 3 columns, each of 35 units in the same line:
    "Return to the start of program:" and "Exit the program:"
    In the second line, use the same format to print:
    "Enter 1" and "Enter 0"

    Prompt for input of integer variable y
    If y is not an integer then:
    Clear input buffer memory for y
    Print "Invalid selection! You will now be returned to the start of
    program"
    Assign 1 to y

    If y is not 1 and y is also not 0:
    Print "Invalid selection! You will now be returned to the start of
    program"
    Assign 1 to y

Endpoint of main loop, the loop may repeat or end which will also end
the program based on the input for y.

Pass out: integer 0

End of Main Function
```

Step 4 - Code or implementation:

TrafficLight.h Header file:

```
#pragma once
#include <iostream>
#include <cmath>
#include <fstream>
#include <ctime>

using namespace std;

class TrafficLight {
private:
	int ID;
	int state; //0 for off, 1 for red, 2 for yellow, and 3 for green
	double greentime;
	static int TLnum; //To record number of instances of this class

public:
```

```
        //Constructor
        TrafficLight() {
                ID = TLnum;
                state = 1; //Every instance starts as red
                greentime = 0;
                TLnum++;
        }

        //Getter functions
        int getID() {
                return (ID);
        }
        int getState() {
                return (state);
        }
        double getGreentime() {
                return (greentime);
        }
        //Setter functions
        void setState(int color) {
                state = color;
        }

        void setGreentime(double GreenTime) {
                greentime = GreenTime;
        }

        void PrintTLinfo() {
                cout << "ID = " << ID << endl;
                cout << "State = ";
                if (state == 1) cout << "Red" << endl;
                if (state == 2) cout << "Yellow" << endl;
                if (state == 3) cout << "Green" << endl;
                if (state == 0) cout << "Off" << endl;
                cout << "Green-time = " << greentime << endl;
                cout << endl;
        }
        static int get_numofTL() {
                return TLnum;
        }
        void wait(double seconds) {
                clock_t start = clock();
                double wait = seconds * CLOCKS_PER_SEC;
                while (clock() < (start + wait)); //Elapse the wait time
        }
};

int TrafficLight::TLnum = 0; //Initializing TLnum
```

Intersection.h Header file:

```
#pragma once

#include <iostream>
#include <cmath>
#include <fstream>
#include <ctime>
```

```
#include <iomanip>
#include "TrafficLight.h"

using namespace std;

string filename; //For input of file name of data file
//Defining constants as global variables
const int x = 1; //Counter for run()
const double yellowtime = 3;  //3 seconds
const double updatetime = (1.5 * 60.0); //Number of seconds in 1.5 minutes, short
interval for test purposes
const int TLmax = 5; //Maximum traffic lights in intersection

class Intersection {
private:
	TrafficLight TL[TLmax]; //Array of TrafficLight instances
	double C; //Cycle length
	double Q_t; //Total traffic flow
	double Q_i[TLmax]; //Traffic flow for traffic light at index TLmax
	double green_time[TLmax]; //Array of greentimes to be stored in TrafficLight
instances
	int TLnum; //To trakc total number of TrafficLight instances employed in
Intersection instance

public:
	//Constructor function
	Intersection() {
		TLnum = 0;
		C = 0;
		Q_t = 0;
		for (int i = 0; i < TLmax; i++) {
			//Set traffic flows and greentimes to 0
			Q_i[i] = 0;
			green_time[i] = 0;
		}
	}

	void AddLight() {
		if (TLnum < TLmax) {
			TrafficLight a;
			TL[TLnum] = a;
			TLnum++;
		}
		else
			cout << "\nLimit reached! Cannot add more traffic lights" << endl;
	}
	void droplight(int TL_ID) {
		bool isTL = false;

		for (int i = 0; i < TLnum; i++)
		{
			if (TL_ID == TL[i].getID())
			{
				for (int j = i; j < TLnum; j++)
				{
					TL[j] = TL[j + 1];
				}
				TLnum--;
```

```
                        cout << "Removed traffic light " << TL_ID << endl;
                        isTL = true;
                    }
                }

                if (!isTL)
                    cout << "Can't find the traffic light to drop!" << endl;
            }

            int get_numofTL() {
                return TLnum;
            }


            void readTrafficData() {

                Q_t = 0; //Initiallziing total traffic flow for every time this function is
called
                ifstream infile; //Input filestream

                //Validate input of filename
                while (int z = 1) {
                    infile.open(filename, ios::in);

                    if (infile.fail()) { //Check if file opened successfully
                        cerr << "Error opening file!\nPlease enter the correct file
name: \n";
                        getline(cin, filename);
                    }
                    else break;
                }

                //Storing data into variables
                string skip;
                getline(infile, skip, '='); //Skip to data
                infile >> C;
                //Subtracting yellowtime from C to get greentimes only, useful in
calculations later
                C -= TLnum * yellowtime;

                for (int i = 0; i < TLnum; i++) { //Store respective traffic flows
                    if (infile.eof()) break;
                    getline(infile, skip, '=');
                    infile >> Q_i[i];
                    Q_t += Q_i[i]; //Calculating total traffic flow as the sum of all
traffic flows
                }

                infile.close(); //Close input file
            }

            void calculate_greentime() {
                for (int i = 0; i < TLnum; i++) {
                    //Calculating greentime
                    double greentime = ((Q_i[i] / Q_t) * C);
                    green_time[i] = greentime;
                    //Setting value of greentime
                    TL[i].setGreentime(green_time[i]);
```

```
                    cout << "\nGreen Time for Traffic Light " << (i + 1) << " is = " <<
TL[i].getGreentime() << " seconds" << endl;
                }
        }

        void run() {
                double SimulationT = 0;
                int cycle = 1;
                cout << "\nCommencing Simulation\n" << endl;

                while (x == 1) { //Loop does not end
                        double start = 0;
                        double end = 0;
                        double time_Cycle = 0;
                        start = clock(); //Start time
                        cout << "Cycle " << cycle << endl << endl;
                        cycle++;

                        for (int i = 0; i < TLnum; i++) {

                                TL[i].setState(3); //Set state to green
                                cout << "Traffic Light " << (i + 1) << ":\nGreen\n";
                                TL[i].wait(TL[i].getGreentime()); //To elapse greetime
                                TL[i].setState(2); //State becomes yellow
                                cout << "Yellow\n";
                                TL[i].wait(yellowtime); //To elapse yellowtime
                                TL[i].setState(1); //State becomes red
                                cout << "Red\n" << endl;

                        }
                        end = clock(); //End time of cycle
                        time_Cycle = end - start; //Time for cycle
                        SimulationT += time_Cycle; //Time in simulation

                        cout << "Time elapsed since start of simulation: " << (SimulationT /
CLOCKS_PER_SEC) << " seconds\n" << endl;

                        if (SimulationT >= (updatetime * CLOCKS_PER_SEC)) { //Check if total
time elapsed is greater than or equal to updatetime
                                updateTiming(); //Update green times from new sensor info in
data file
                                cout << "\nThe time for the updated simulation will now be
reset\n" << endl;
                                SimulationT = 0;
                                cycle = 1;
                        }
                        //Now cycles restart
                }
        }

        void updateTiming() {
                cout << "\nGreen times for all traffic lights will now be updated\n";
                readTrafficData(); //Reads new data stored by sensors
                calculate_greentime(); //To update greentimes
        }
};
```

Source.cpp file:

```
/*-----------------------------------------------------------------------------------------
*/
/* Name: Muhammad Zaeem Shahzad, Student ID: ms12297 */
/* Date: November 27, 2020 */
/* Program: assignment4.cpp */
/* Description: Traffic Light Control System: This program simulates the control of
traffic
lights at an intersection, displaying their characteristic details at set time intervals
*/
/*-----------------------------------------------------------------------------------------
*/

#include <iostream> //For input/output
#include <fstream> //For file-handling
#include <iomanip> //For output manipulation eg: in tabular format
#include<sstream> //For strings' content handling eg: getline function
#include <cmath> //For complex calculations
#include<ctime> //For use of time functions
#include "TrafficLight.h" //Contains the TrafficLight class
#include "Intersection.h" //Contains the Intersection class

using namespace std;

int main() {

	Intersection road; //Intersection instance
	int TLnum; //Number of traffic lights user wants to add
	int y = 1; //Counter for loops

	cout << "Welcome to the Traffic Light Control System Simulator!\n";
	cout << "You can run a simulation of traffic lights at an intersection in this
program\n";
	cout << "Note that the program supports a maximum of 5 traffic lights at the
intersection\n";

	while (y == 1) {
		cout << "\nEnter the number of traffic lights (not more than 5) you want to
simulate at the intersection: \n";
		cout << "You may enter 0 to exit program\n";

		if (!(cin >> TLnum)) {
			//To clear the buffer memory
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');

			cout << "Please enter a positive integer!\n";
			continue;
		}

		else {
			if (TLnum > 5) {
				cout << "You must enter an integer value less than 5!\nTry
again\n";
				continue;
			}
```

```
                        if (TLnum <= 0) {
                                cout << "You chose to exit, please confirm\n";
                                y = 0;
                        }
                        if (TLnum > 0 && TLnum <= 5) {
                                for (int i = 0; i < TLnum; i++) {
                                        road.AddLight();
                                }
                                cout << "Enter the name of the file that contains the traffic
information stored by the sensors: \n";

                                //Clearing buffer memory for input stream
                                cin.clear();
                                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                                //Prompting input of filename
                                getline(cin, filename);

                                road.readTrafficData(); //Read data from input file
                                road.calculate_greentime(); //Calculate greentimes for all
traffic lights
                                road.run(); //Run simulation
                        }
                }

                //Exit display
                cout << "\nTo " << endl;
                cout << left << setw(35) << "Return to the start of program:" << setw(35)
<< "Exit the program:" << endl;
                cout << left << setw(35) << "Enter 1" << setw(35) << "Enter 0\n" << endl;

                //Validating counter for the return to start loop
                //To ensure that y can only be 1 or 0

                if (!(cin >> y)) { //To prompt for input of y AND validate an integer input

                        //To clear the buffer memory
                        cin.clear();
                        cin.ignore(numeric_limits<streamsize>::max(), '\n');

                        cout << "\nInvalid selection!\nYou will now be returned to the start
of program\n" << endl;
                        y = 1;
                }

                if (y != 1 && y != 0) { //To validate input of only 1 or 0
                        cout << "\nInvalid selection!\nYou will now be returned to the start
of program\n" << endl;
                        y = 1;
                }

        }

        return 0;
}
```

Step 5 - Test and Verification:

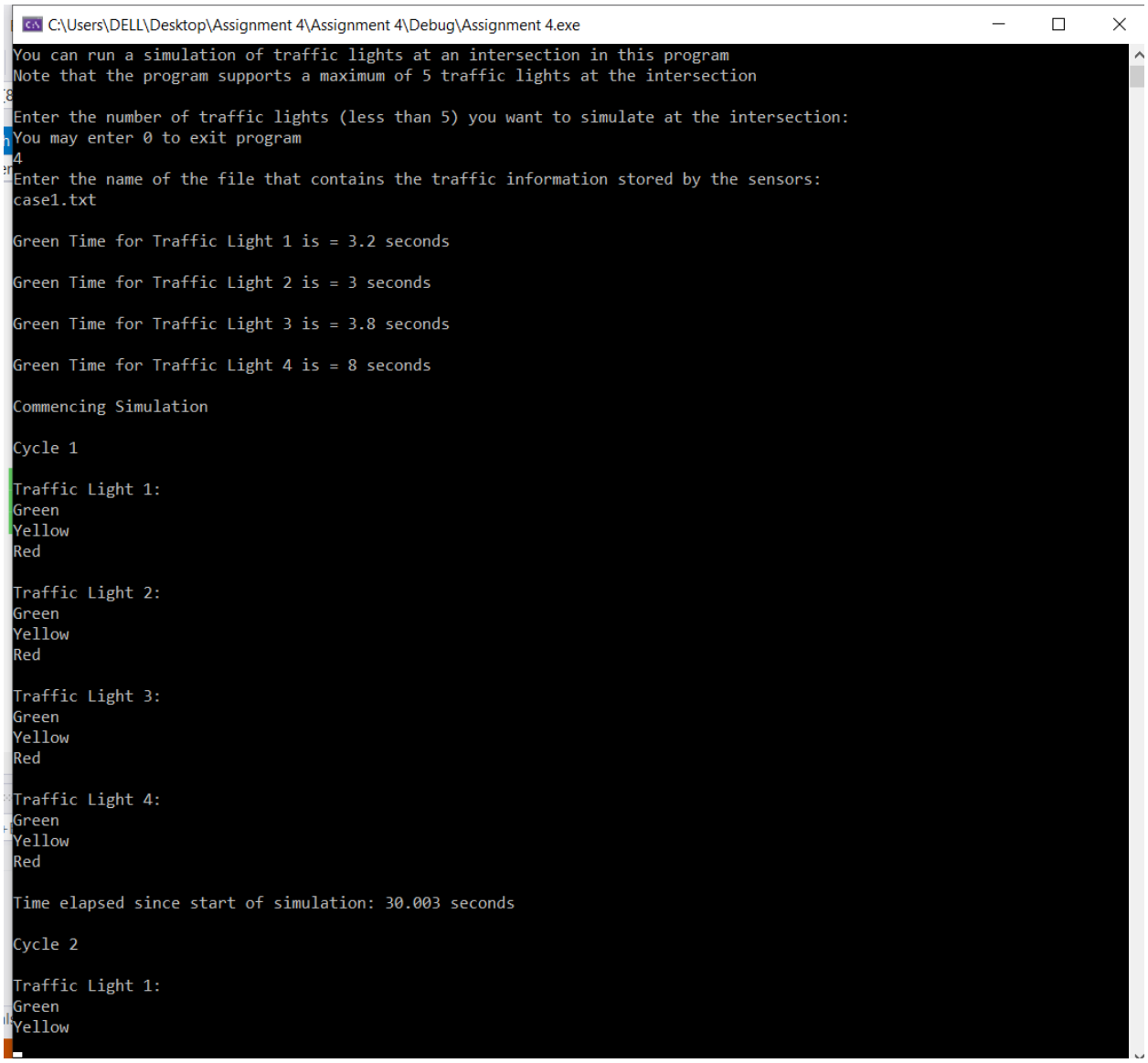
- *Test Case 1 – Using 4 traffic lights and input file case1.txt:*

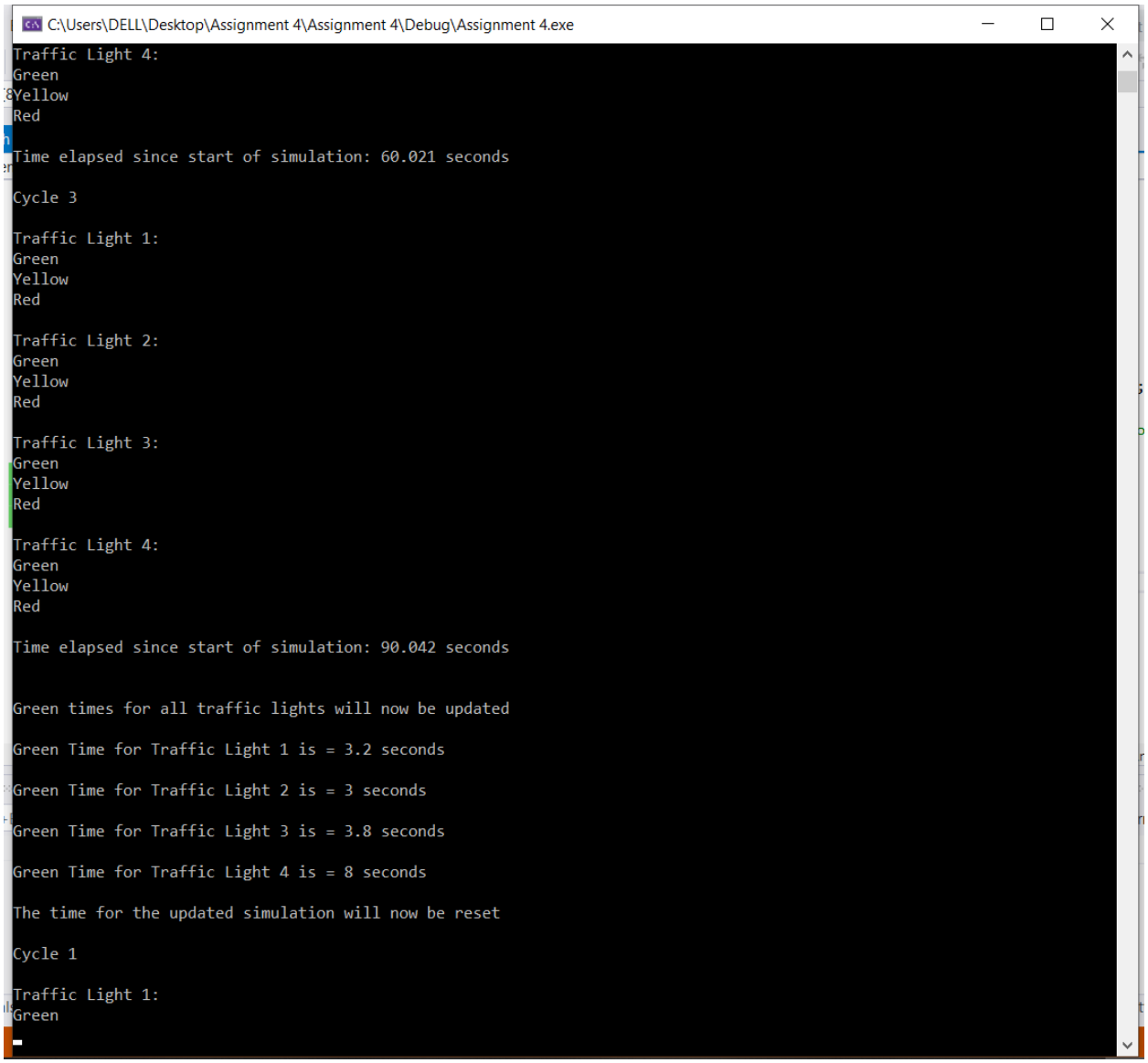
    - *For Inputs:*
        - *4*
        - *case1.txt*

    - *Data entered into the program:*
        - *4 traffic lights at the intersection*
        - *Traffic information input file name entered*

    - *The Outputs were:*
        - *Simulation of 4 traffic lights with inputs from case1.txt*

Console Window for the result of test case 1:


C:\Users\DELL\Desktop\Assignment 4\Assignment 4\Debug\Assignment 4.exe
You can run a simulation of traffic lights at an intersection in this program
Note that the program supports a maximum of 5 traffic lights at the intersection
Enter the number of traffic lights (less than 5) you want to simulate at the intersection:
You may enter 0 to exit program
4
Enter the name of the file that contains the traffic information stored by the sensors:
case1.txt
Green Time for Traffic Light 1 is = 3.2 seconds
Green Time for Traffic Light 2 is = 3 seconds
Green Time for Traffic Light 3 is = 3.8 seconds
Green Time for Traffic Light 4 is = 8 seconds
Commencing Simulation
Cycle 1
Traffic Light 1:
Green
Yellow
Red
Traffic Light 2:
Green
Yellow
Red
Traffic Light 3:
Green
Yellow
Red
Traffic Light 4:
Green
Yellow
Red
Time elapsed since start of simulation: 30.003 seconds
Cycle 2
Traffic Light 1:
Green
Yellow



The results are as expected.

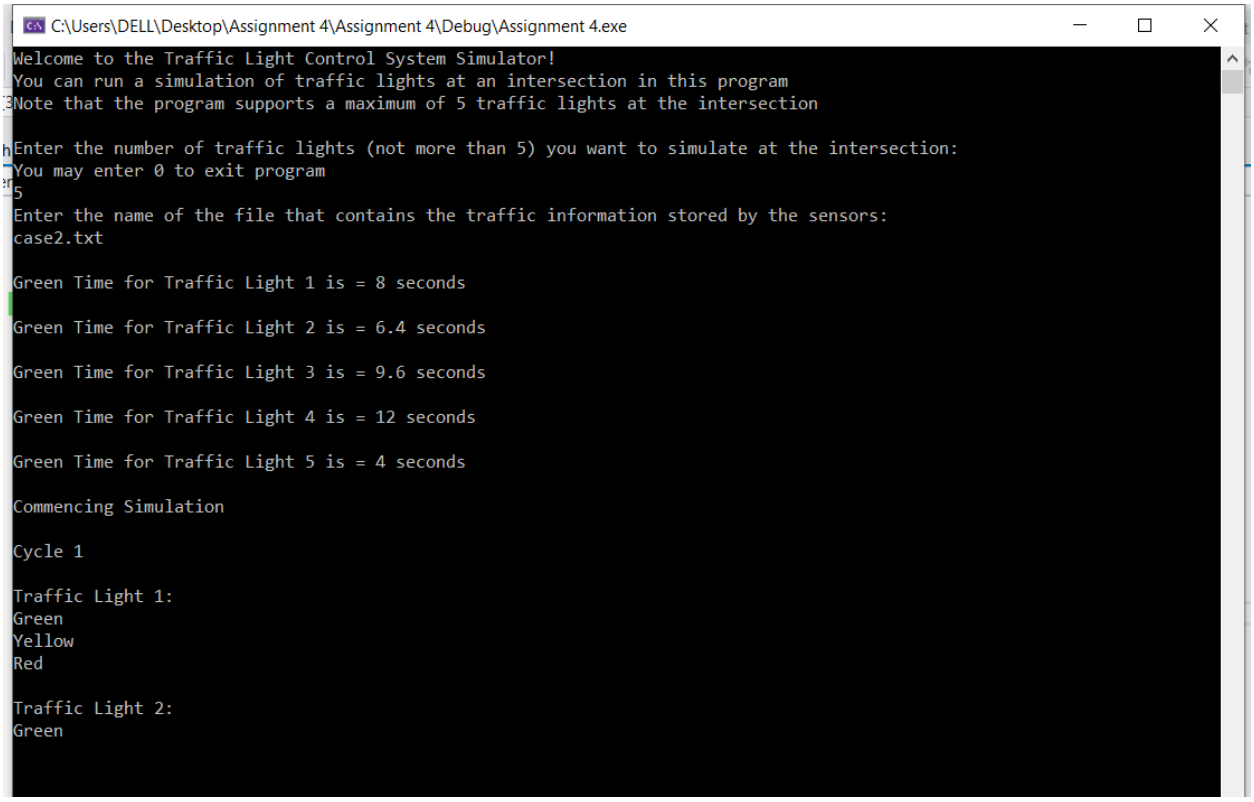
- *Test Case 2 – Using 5 traffic lights and input file case2.txt:*

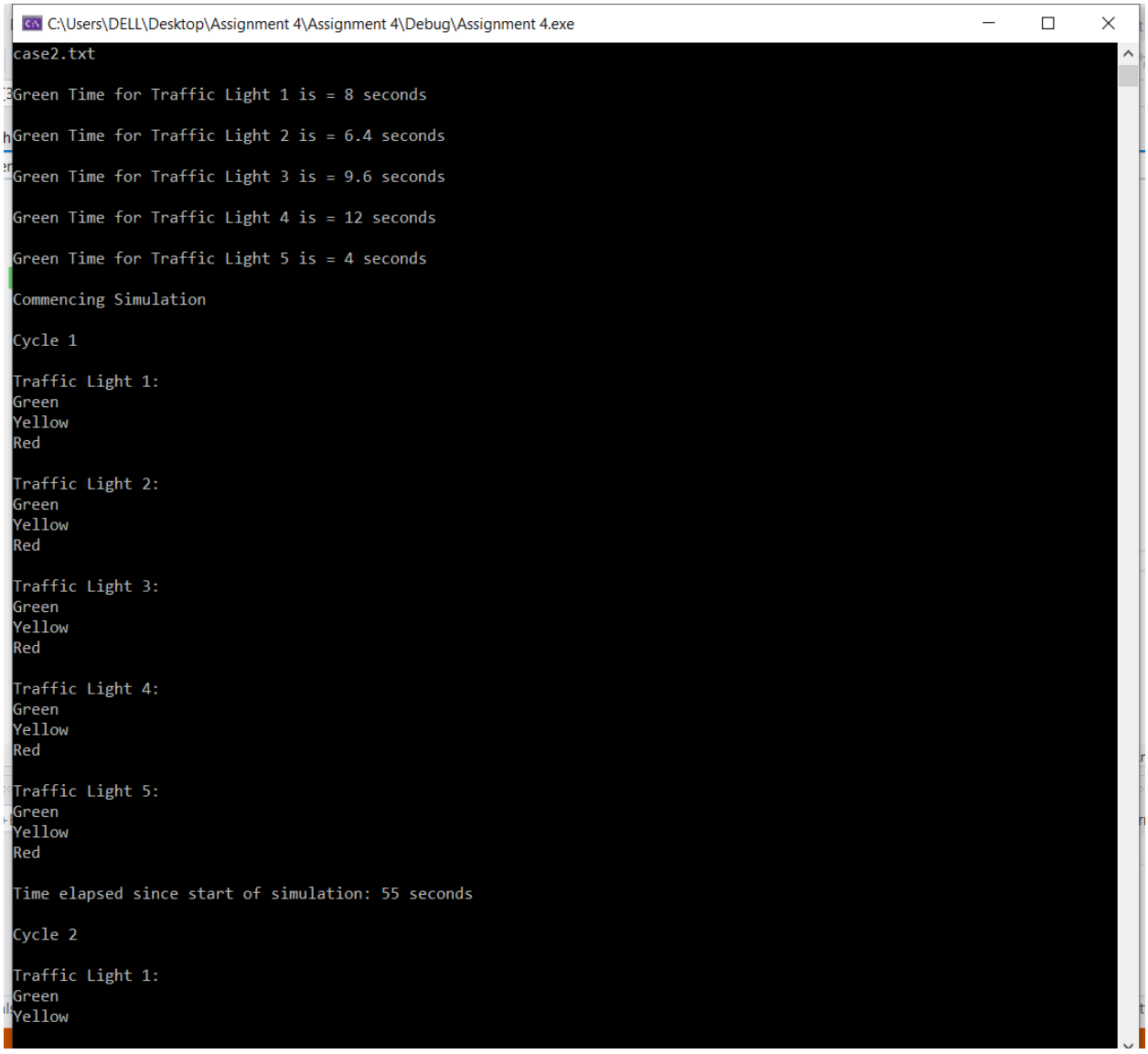
  - *For Inputs:*
    - *5*
    - *case2.txt*

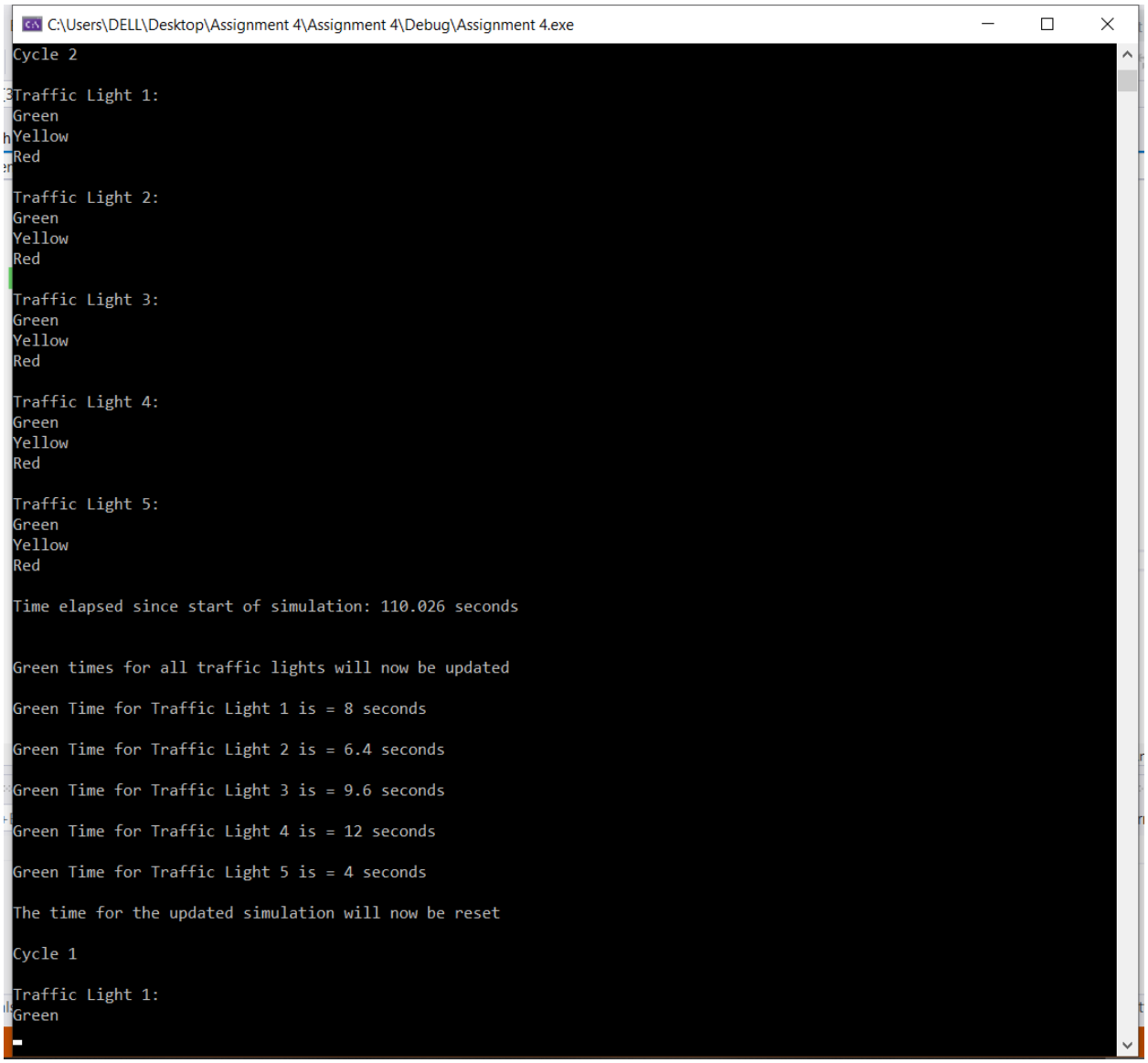
  - *Data entered into the program:*
    - *5 traffic lights at the intersection*
    - *Traffic information input file name entered*

- *The Outputs were:*
  - *Simulation of 5 traffic lights with inputs from case2.txt*

Console Window for the result of test case 2:




C:\Users\DELL\Desktop\Assignment 4\Assignment 4\Debug\Assignment 4.exe
case2.txt
Green Time for Traffic Light 1 is = 8 seconds
Green Time for Traffic Light 2 is = 6.4 seconds
Green Time for Traffic Light 3 is = 9.6 seconds
Green Time for Traffic Light 4 is = 12 seconds
Green Time for Traffic Light 5 is = 4 seconds
Commencing Simulation
Cycle 1
Traffic Light 1:
Green
Yellow
Red
Traffic Light 2:
Green
Yellow
Red
Traffic Light 3:
Green
Yellow
Red
Traffic Light 4:
Green
Yellow
Red
Traffic Light 5:
Green
Yellow
Red
Time elapsed since start of simulation: 55 seconds
Cycle 2
Traffic Light 1:
Green
Yellow



The results are as expected.

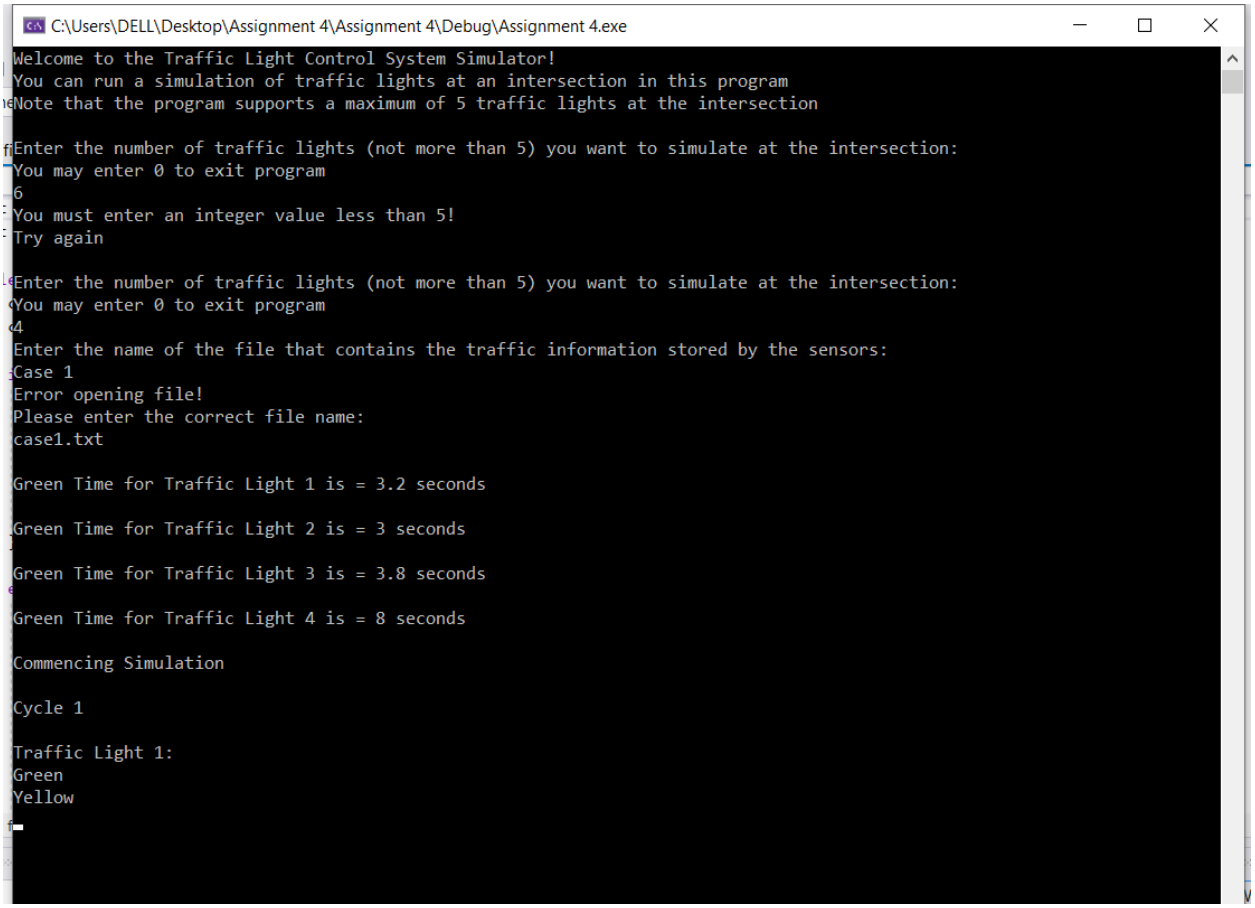
- *Test Case 3 – Input Validation:*

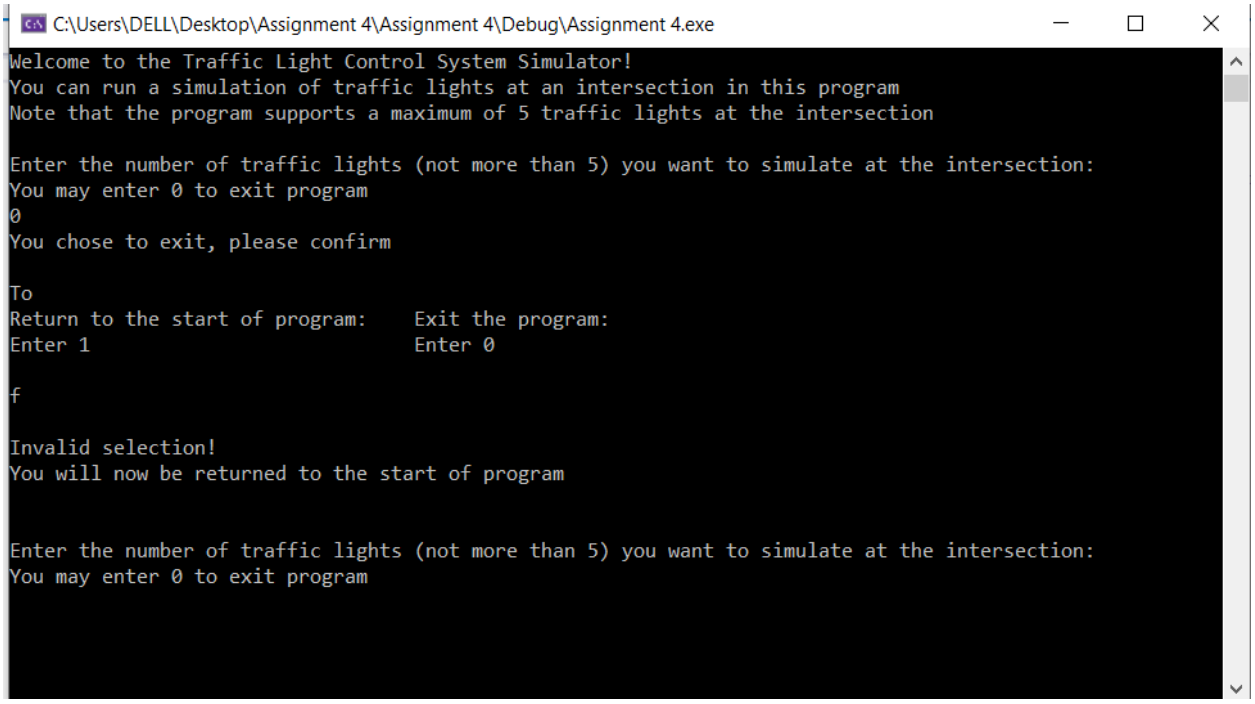
  - *For invalid inputs:*
    - *6 for traffic light number*
    - *Case1 for file name*
    - *f in Exit Display*

  - *The Outputs were:*
    - *Error message and prompt for re-input for traffic light number*
    - *Error message and prompt for re-input for file name*

- *Restart of the program*

Console Window for the result of test case 3:





The results are as expected.

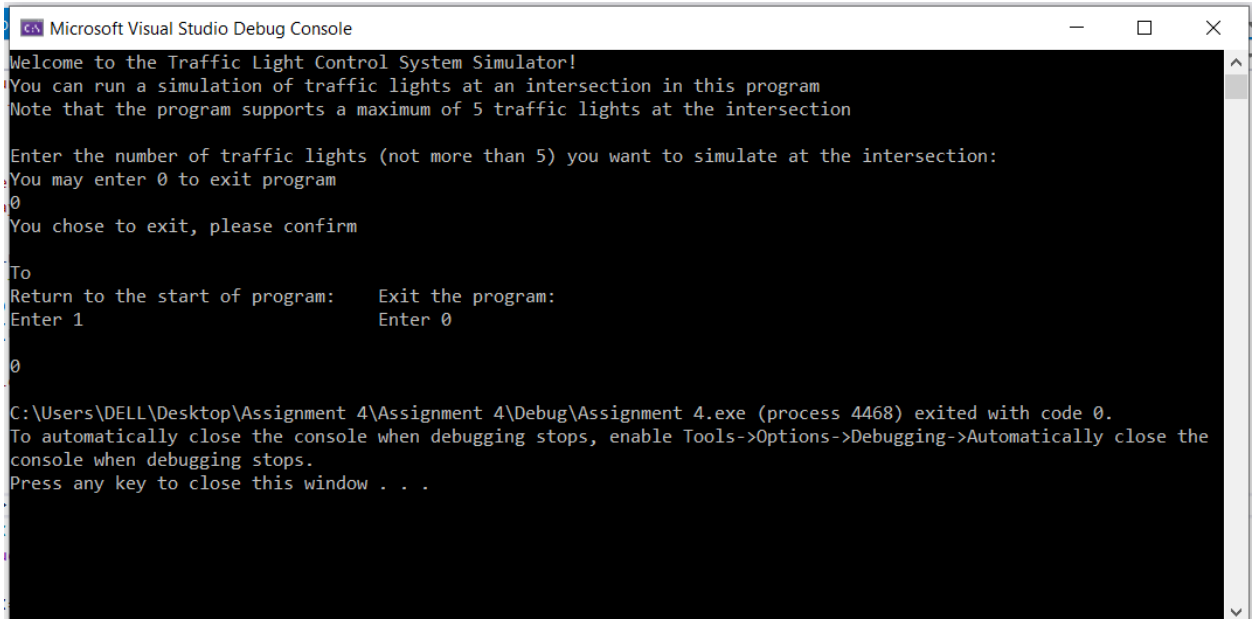
- *Test Case 4 – Termination of Program after input of 0 in traffic light number:*

  - *For Inputs:*
    - *0*
    - *0*

  - *Data entered into the program:*
    - *Exit program option chosen by entering 0 traffic light number*
    - *Exit Program chosen from Exit Display*

  - *The Outputs were:*
    - *Confirmation Message*
    - *Exit Display*
    - *Termination of Program*

Console Window for the result of test case 4:



The result is as expected.