Faculty of Computer & Information Sciences
Ain Shams University
Subject: CSW150
Fundamentals of Structured Programming

Examiners: Prof. Zaki Taha
Dr. Yasmine Afify
Dr. Salsabil Amin
Academic year: 2nd term 2019-2020
Year: 1st undergraduate

Research Topic Version (D)

Title: Traffic Control System

تحذير هام: علي الطالب عدم كتابة اسمه أو كتابة اي شيء يدل علي شخصيته

1. Data Model

1.1, 1.2

const int kMax_fines_number = 100, kRoad_maximum_speed = 70, kNumber_of_drivers = 10, kMax_cars_number = 3;

The maximum allowed number of cars each driver can own is 3, and I assumed that one driver can't have more than 100 fines at a time, the maximum allowed road speed is 70 km/h, and there are 10 drivers registered in this system. 3, 100, 70, 10 are integers. Their values are fixed so I defined them as constants.

Date, Fine, Car, Driver, and Update: I used struct because each one of them stores more than one variable of different types. For example: a date has a day and a month and a year. I should know value, street name, date and status (paid/not) for each fine. The system stores the car plate number, model, list of recorded fines and production year for each registered car. I should know license number, person name, birthdate and list of owned cars and total unpaid fines value for each driver. For each new update I should know that update is for which driver, which of this driver's cars and for which of the assigned fines.

int day, month, year; day's maximum value can be 31, month's maximum value can be 12, and year. They are all integers.

They don't have a fixed value so they are variables (birthdate change from one driver to another and the date of assigning a fine is an input from the user).

int value; maximum value of a fine can reach 760, which is an integer, because I assume that the maximum allowed road speed is 70 km/h and the maximum

speed a car can ever reach is 450 km/h. Double the difference is equal to 760. It's a variable because it depends on the speed of a car.

string status, steet_name, model, name;

stasus (paid or unpaid), a driver's name, a car model are texts so I defined them as strings. status can be "Paid." Or "Unpaid.", street name is an input from the user, model varies from a car to another, and name varies from a driver to another or it's an input from the user when paying a fine or searching for information, so they are variables.

int plate_number; I assume that a valid plate number is a 3-digit number which is an integer. A plate number varies from a car to another or it's an input from the user when assigning a fine, paying a fine, or searching for a car, so they are variables.

Fine fines[kMax_fines_number]; is an array of structures because it contains more than one structure (the same structure). kMax_fines_number is the size of the array because I assumed the maximum number of fines assigned to car can reach 100 as mentioned before. It's a variable array because each fine has different data (value, street name, status, and date).

int production_year; because a year is an integer. It varies from one car to another so it's a variable.

int total_unpaid; I mentioned before that a maximum fine value can be 760 and a driver can have a maximum of 3 cars. Each car can have a maximum of 100 fines at a time. That means that one driver can have a maximum of 228000 total unpaid fine value at a time which is an integer. Total unpaid fines varies from one driver to another, so it's a variable.

int license_number; I assumed a license number is a 9-digit number so it's an integer. It's a variable because it varies from a driver to another, or it's an input from the user when paying a fine.

birthdate is a structure (Date) because it has a day, a month and a year.

Car cars[kMax_cars_number]; is an array of structures because it contains more than one structure (the same structure). Its size is kMax_cars_number because each driver can own a maximum of 3 cars as mentioned before. It’s a variable array because data of each car varies from a car to another.

int which_driver; It can’t exceed 10, so it’s an integer. It doesn’t have a constant value so it’s a variable (it depends on which driver is paying a fine).

int which_car; It can’t exceed 3, so it’s an integer. It doesn’t have a constant value so it’s a variable (it depends on for which car a driver is paying a fine).

int which_fine; It can’t exceed 100, so it’s an integer. It doesn’t have a constant value so it’s a variable (it depends on which of the assigned fines is being paid).

Driver drivers[kNumber_of_drivers]; is an array of structures because it contains more than one structure (the same structure). Its size is kNumber_of_drivers because number of registered drivers is 10. It’s a variable array because each driver has different data (license number, plate number, cars data, total unpaid fines value, name, and birthdate).

Update updates[3005]; is an array of structures because it contains more than one structure (the same structure). 3005 is its size because we have 10 drivers, each one of them can have 3 cars, and each car of them can have 100 fines. So the total number of fine payment process can reach 3000 (10*100*3). I added 5 just for safety. Each update is for a different data (which driver is the update for? Which of his cars? Which of the assigned fines?), so it’s a variable array.

bool hasFoundCar; because there’ll be a car matches the entered data or not (true or false) , so it’s a Boolean. It can have two values (true or false) so it’s a variable.

Date fine_assignment_date; any date can be represented as a structure as I mentioned before. It’s an input from the police officer, so it’s a variable.

int car_speed; I assumed the maximum speed a car can reach is 450 km/h which is an integer. Speed varies from a car to another and it's an input from the police officer, so it's a variable.

bool hasFoundFine; when a driver choose to pay a fine, there can be an unpaid fine assigned to him or not (true or false), so it's a Boolean. It can have two values (true or false), so it's a variable.

int counter; (in PayByPlate function) can reach a maximum of 100 which is the number of fines assigned to a specific car, so it's an integer. It counts the number of fines assigned to a specific car, its value isn't constant; so it's a variable, (in PayByName function) can reach a maximum of 300, which is an integer, because one driver can have 3 cars, and each car can have 100 fines. It counts the number of fines assigned to a specific driver, its value isn't constant, so it's a variable, and (in PayByLicense function) can reach a maximum of 300, which is an integer, because one driver can have 3 cars, each car can have 100 fines. It counts the number of fines assigned to a specific driver, its value isn't constant, so it's a variable.

int fine_choice; (in PayByPlate function) The number of the fine assigned to a specific car can reach a maximum of 100, so it's an integer. It's an input from the driver, so it's a variable, (in PayByName function) the number of the fine a driver want to pay can reach a maximum of 300, so it's an integer. It's an input from the driver, so it's a variable, and (in PayByLicense function) the number of the fine a driver want to pay can reach a maximum of 300, so it's an integer. It's an input from the driver, so it's a variable.

int payment_method; there are 3 methods, which is an integer, to pay a fine (by name, plate number, or license number). It's an input from the driver, so it's a variable.

int search_choice; You can search for information by a driver's name, or a car plate number, so the number of choices (2) is an integer. It's an input from the driver, so it's a variable.

int use_again; you can choose between yes or no, so the number of choices is 2 which is an integer. It’s an input from the driver, so it’s a variable.

int service_choice; there are 3, which is an integer, services (assign a fine, pay a fine, show information). It’s an input from the user, so it’s a variable.

---

const int kMax_fines_number = 100, kRoad_maximum_speed = 70, kNumber_of_drivers = 10, kMax_cars_number = 3;

int plate_number, counter, fine_choice;

bool hasFoundCar, hasFoundFine;

Driver drivers[kNumber_of_drivers];

Update updates[3005];

These are global because they are used in more than one function. The others are local because they are only used in their scopes.

1.3

- At main menu, a user can enter 1 to assign a fine to a driver, 2 to pay a fine, 3 to search for information. Any other choice will be invalid, a message will be displayed saying that it’s an invalid choice.
- A police officer isn’t allowed to enter a speed more than 450 km/h, a message will be displayed saying that a car speed can’t exceed 450 km/h and he’ll have to enter another valid speed.
- A police officer isn’t allowed to enter a negative speed, a message will be displayed saying that a car speed can’t be negative and he’ll have to enter another valid speed.
- A police officer isn’t allowed to enter a wrong date (a day more than 31 or less than 1, a month more than 12 or less than 1) when assigning a fine, a message will be displayed and he’ll have to enter a valid date.

- A police officer can’t assign a fine to a car that isn’t registered to the system, a message will be displayed saying that there’s no such car registered to the system if he entered a wrong plate number.
- A police officer can’t assign a fine to a car that didn’t exceed the road maximum speed. A message will be displayed saying that no fine has been assigned because the car speed didn’t exceed the allowed road speed.
- When a driver wants to pay a fine he can enter 1 to pay by his name, 2 to pay by plate number, or 3 to pay by his license number. If he entered another number a message will be displayed saying that it’s an invalid choice.
- A driver can’t choose to pay a fine assigned to a car that isn’t registered to the system, a message will be displayed saying that there’s no such car registered to the system if he entered a wrong plate number. Also, when writing a right plate number, a list of fines appears for him to choose between, he can’t choose a wrong choice. If he chose a fine number that isn’t displayed on the fines list, a message will be displayed saying that it’s an invalid choice.
- A driver who isn’t registered to the system can’t pay a fine by his name or by his license number, a message will be displayed saying that there’s no such driver registered to the system.
- A user can’t search for information about a driver (by name) who isn’t registered to the system, a message will be displayed saying that there’s no such driver.
- A user can’t search for information about a car (by plate number) that isn’t registered to the system. A message will be displayed saying that there’s no such car.

- When the user choose to search for information, he can enter 1 to search by name or 2 to search by plate number, if he didn't enter 1 or 2 a message will be displayed saying that it's an invalid choice.
- The user is asked whether he wants to go back to the main menu or not each time he uses the system. He can enter 1 to choose yes, or 2 to choose no. If he entered any other number a message will be displayed saying that it's an invalid choice and he'll be forced to enter a valid choice.

2. Logical Model (Algorithm)

```
FUNCTION AssignFine()
    PRINT "Note: Maximum allowed road speed is 70 km/h."
    PRINT new line
    SET hasFoundCar = false
    PRINT "Street name: "
    READ street_name
    DO
        PRINT new line
        PRINT "Car speed: "
        READ car_speed
        IF car_speed > 450 THEN
            PRINT "Speed can't exceed 450 km/h."
        ELSE IF car_speed < 0 THEN
            PRINT "Speed can't be a negative number."
            PRINT newline
        ENDIF
```

```
 WHILE car_speed > 450 OR car_speed < 0
 ENDWHILE
PRINT new line
PRINT “Car plate number: “
READ plate_number
DO
     PRINT new line
     PRINT “Date: Day)  “
     READ fine_assignment_date.day
     IF fine_assignment_date.day <= 0 OR fine_assignment_date.day > 31
     THEN
            PRINT “Day must be a number from 1 to 31.”
     ENDIF
 WHILE  fine_assignment_date.day <= 0 OR fine_assignment_date.day > 31
 ENDWHILE
 DO
     PRINT “Month)  “
     READ fine_assignment_date.month
     IF fine_assignment_date.month <= 0 OR fine_assignment_date.month > 12
     THEN
            PRINT “Month must be a number from 1 to 12.”
            PRINT new line
     ENDIF
```

```
WHILE  fine_assignment_date.month <= 0 OR fine_assignment_date.month >
12
ENDWHILE
PRINT "Year) "
READ fine_assignment_date.year
IF car_speed <= kRoad_maximum_speed THEN
   PRINT "No fine has been assigned, The Car speed didn't exceed maximum
   allowed road speed."
   PRINT new line
ELSE
  FOR (drivers_counter = 0, drivers_counter < kNumber_of_drivers,
         drivers_counter + 1)
     FOR (cars_counter = 0, cars_counter < kMax_cars_number, cars_counter
+ 1)
     FOR (fines_counter = 0, fines_counter < kMax_fines_number,
              fines_counter + 1)
          IF drivers[drivers_counter].cars[cars_counter].plate_number =
               plate_number
             AND drivers[drivers_counter].cars[cars_counter].
                fines[fines_counter].value = 0
             SET hasFoundCar = true
             SET drivers[drivers_counter].cars[cars_counter].
                fines[fines_counter].value =
                (car_speed - kRoad_maximum_speed) * 2
             SET drivers[drivers_counter].cars[cars_counter].
                fines[fines_counter].date = fine_assignment_date;
                SET drivers[drivers_counter].cars[cars_counter]
                    .fines[fines_counter].status = “Unpaid.”
                SET drivers[drivers_counter].cars[cars_counter].
                    fines[fines_counter].steet_name = street_name
                drivers[drivers_counter].total_unpaid =
                    drivers[drivers_counter].total_unpaid  +
                    (car_speed - kRoad_maximum_speed) * 2
                PRINT “Done !”
                PRINT new line
                RETURN
              ENDIF
          ENDFOR
        ENDFOR
      ENDFOR
        IF hasFoundCar = false THEN
              PRINT “There is no car with this plate number.”
          PRINT new line
        ENDIF
    ENDIF
ENDFUNCTION
```

3. Process Model (Functions)

- void SystemData(); It’s a void function. It takes no input and returns no output. It just stores all the data about all the drivers registered to the system, their name (string), birthdate (structure), car model (string), plate number, license number (integer), and production year (integer).
- void AssignFine(); It’s a void function. It takes no input and returns no output. But in its body it takes input from the user. It takes street_name (a

string), car_speed (an integer), plate_number (an integer), and fine_assignment_date (a structure). Then, it searches for the car with this plate number, and assign all the fine information (value (integer), date (structure), street name (string), and status (string)) to it. It does nothing if the car speed didn't exceed the maximum allowed road speed, or it says that there's no such car if there wasn't a car with this plate number registered to the system.

- Void PayByPlate(); It's a void function. It takes no input and returns no output. But in its body it takes input from the user. It takes plate_number (an integer), searches for the car with this plate number and says that there are no unpaid fines assigned to this car or displays its unpaid fines list (value(integer), date (structure), street name(string)), the user enters the number (integer) of the fine we wants to pay. If he entered a valid choice a message says "done !" will appear, otherwise an error message saying "invalid choice !" will appear. If the user entered a wrong plate number nothing will be done and a message will appear saying that there's no such car registered to the system.
- Void PayByName(); It's a void function. It takes no input and returns no output. But in its body it takes input from the user. It takes name (string), searches for cars that this driver owns and says that there's no unpaid fines assigned to this driver, or displays each of the cars (it displays each car model (string)) unpaid fines list (value(integer), date (structure), street name(string)), the user enters the number (integer) of the fine he wants to pay. If he entered a valid choice a message says "done !" will appear, otherwise an error message saying "invalid choice !" will appear. If the user entered a wrong name, nothing will be done and a message will appear saying that there's no such driver registered to the system.

- Void PayByLicense(); It’s a void function. It takes no input and returns no output. But in its body it takes input from the user. It takes license_number (an integer), searches for the driver with this license number and says that there’s no unpaid fines assigned to this driver or displays each of his cars (it displays each car model (string)) unpaid fines list (value(integer), date (structure), street name(string)), the user enters the number (integer) of the fine we wants to pay. If he entered a valid choice a message says “done !” will appear, otherwise an error message saying “invalid choice !” will appear. If the user entered a wrong license number nothing will be done and a message will appear saying that there’s no such driver registered to the system.
- Void PayFine(); it’s a void function. It takes no input and returns no output. But in its body it takes input from the user. It displays methods by which user can choose to pay a fine (plate number, name, or license number) and the user enters the number of his choice (integer). Based on his choice the right function will be called (PayByPlate, PayByName, or PayByLicense) but if the user entered a wrong choice a message will appear saying that it’s an invalid choice.
- Void SearchByName(); It’s a void function. It takes no input and returns no output. But in its body, it takes input from the user. It takes name (string). Then, it searches for the driver with this name and displays all his information (total unpaid fines (integer), his birthdate (structure), his license number (integer), each of his cars model (string), plate number (integer), production year (integer), each of assigned fines value (integer), street(string), date(structure), and status (string), or says that there are no assigned fines), or says that there’s no such driver registered to the system in case of entering a wrong name.

- Void SearchByPlateNumber(); It’s a void function. It takes no input and returns no output. But in its body, it takes input from the user. It takes plate_number (integer). Then, it searches for the car with this plate number and displays all information about its owner (his name (string), total unpaid fines (integer), his birthdate (structure), his license number (integer), the car model (string), plate number (integer), production year (integer), each of assigned fines value (integer), street(string), date(structure), status (string), or says that there are no assigned fines), or says that there’s no such car registered to the system in case of entering a wrong plate number.
- Void ShowInformation(); It’s a void function. It takes no input and returns no output. But in its body, it takes input from the user. It displays 2 different methods by which a user can search for information (name or plate number). The user enters the number of his choice (integer). Based on his choice the right function will be called (SearchByName or SearchByPlateNumber). If the user entered a wrong choice a message will appear saying that it’s an invalid choice.

4. Coding Style

- It’s preferred to declare the loop variable in the initializer part in the loop and avoid using this variable after the loop → lines 155, 156, 157, 182, 183, 186, 230, 233, 234, 278, 281, 282, 343, 349, 364, 387, 388, and 401.
- In a switch case, the default case should always be there and displays an error message if it shouldn’t be reached → lines 333, 432, and 454.
- Variable names should be meaningful to make your code understandable. It should be lowercase letters with underscores as separators between words → line 5, 6, 7, 9, 12, 13, 14, 17, 18, 19, 20, 23, 24, 25, 26, 27, 30, 31, 32, 34, 118, 122, 124, 321, 422, and 439.

- A constant name should have a leading lowercase k which is followed by an uppercase letter → line 5.
- You should #include as little as you can because it consumes compile time performance → lines 1, 2, and 3.
- You should start a Boolean variable name by has, is, or canو so you can easily understand that this variable will give you a Boolean value → Line 7.
- A structure name should start with an uppercase letter and have an uppercase letter for each new word. Lines → 8, 11, 16, 22, and 29.
- When naming a function, you should capitalize each new word. Lines → 35, 119, 176, 222, 271, 322, 319, 337, 383, and 421.
- If you think that a part of your code is tricky, you should write a comment explaining it → lines 190, 238, 286, 350, 353, 359, 366, 397, and 403.
- There should be a space after if and before { but not in the braces (condition). Lines → 184, 187, 231, 235, 261, 279, 283, 309, 344, 360, 365, 389, 402.
- There should be a space after semicolons in for loops. Lines → 155, 156, 157, 182, 183, 186, 230, 233, 234, 278, 281, 282, 343, 349, 364, 387, 388, and 401.
- There shouldn't be a space before colon in a switch case. Lines → 324, 327, 330, 333, 426, 429, 432, 445, 448, 451, and 454.
- There should always be spaces around assignment operators. Lines → 5, 36, 37, 38, 39, 40, 41, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 57, 58, 59, 60, 61, 62, 64, 65, 66, 67, 68, 69, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 91, 92, 93, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 121, 155, 156, 157, 161, 162, 163, 164, 177, 178, 180, 182, 183, 185, 186, 189, 198, 214,

227, 228, 229, 230, 232, 233, 234, 237, 247, 262, 273, 276, 277, 278, 280, 281, 282, 285, 310, 340, 343, 345, 349, 364, 386, 387, 388, 390, and 401.

5. Implementation

```
#include <iostream>
#include <string>
#include <stdlib.h>
using namespace std;
const int kMax_fines_number = 100, kRoad_maximum_speed = 70, kNumber_of_drivers = 10,
kMax_cars_number = 3;
int plate_number, counter, fine_choice;
bool hasFoundCar, hasFoundFine;
struct Date {
	int day, month, year;
};
struct Fine {
	int value;
	Date date;
	string status, steet_name;
};
struct Car {
	int plate_number;
	string model;
	Fine fines[kMax_fines_number];
	int production_year;
};
struct Driver {
	int total_unpaid;
	int license_number;
	string name;
	Date birthdate;
	Car cars[kMax_cars_number];
};
struct Update {
	int which_driver;
	int which_car;
	int which_fine;
};
Driver drivers[kNumber_of_drivers];
void SystemData() {
	drivers[0].license_number = 123456789;
	drivers[0].name = "Fathy Ahmed";
	drivers[0].birthdate = { 10, 2, 1990 };
	drivers[0].cars[0].plate_number = 123;
	drivers[0].cars[0].model = "Ferrari 488 Pista";
	drivers[0].cars[0].production_year = { 2019 };

	drivers[1].license_number = 123457789;
	drivers[1].name = "Wael Ahmed";
	drivers[1].birthdate = { 10, 5, 1995 };
	drivers[1].cars[0].plate_number = 125;
	drivers[1].cars[0].model = "Jeep Wrangler";
	drivers[1].cars[0].production_year = { 2017 };

	drivers[2].license_number = 126656789;
	drivers[2].name = "Yehia Fathy";
	drivers[2].birthdate = { 10, 1, 1999 };
	drivers[2].cars[0].plate_number = 873;
```

```
drivers[2].cars[0].model = "Jeep Grand Cherokee";
drivers[2].cars[0].production_year = { 2011 };

drivers[3].license_number = 223456789;
drivers[3].name = "Omnia Maher";
drivers[3].birthdate = { 8, 2, 1998 };
drivers[3].cars[0].plate_number = 223;
drivers[3].cars[0].model = "MINI Hardtop";
drivers[3].cars[0].production_year = { 2014 };

drivers[4].license_number = 323456789;
drivers[4].name = "Esraa Ramy";
drivers[4].birthdate = { 16, 2, 1995 };
drivers[4].cars[0].plate_number = 129;
drivers[4].cars[0].model = "FIAT 500";
drivers[4].cars[0].production_year = { 2018 };

drivers[5].license_number = 423456789;
drivers[5].name = "Elina Yassen";
drivers[5].birthdate = { 10, 10, 1990 };
drivers[5].cars[0].plate_number = 467;
drivers[5].cars[0].model = "FIAT 500e";
drivers[5].cars[0].production_year = { 2016 };

drivers[6].license_number = 523456789;
drivers[6].name = "peter David";
drivers[6].birthdate = { 11, 12, 1999 };
drivers[6].cars[0].plate_number = 758;
drivers[6].cars[0].model = "FIAT 124 Spider";
drivers[6].cars[0].production_year = { 2017 };

drivers[7].license_number = 623456789;
drivers[7].name = "Sayed emam";
drivers[7].birthdate = { 5, 7, 1990 };
drivers[7].cars[0].plate_number = 923;
drivers[7].cars[0].model = "Toyota 86";
drivers[7].cars[0].production_year = { 2017 };
drivers[7].cars[1].plate_number = 953;
drivers[7].cars[1].model = "Toyota Sequoia";
drivers[7].cars[1].production_year = { 2012 };

drivers[8].license_number = 723456785;
drivers[8].name = "Ammar khalid";
drivers[8].birthdate = { 10, 7, 1993 };
drivers[8].cars[0].plate_number = 133;
drivers[8].cars[0].model = "Toyota RAV4";
drivers[8].cars[0].production_year = { 2020 };
drivers[8].cars[1].plate_number = 533;
drivers[8].cars[1].model = "Nissan Altima";
drivers[8].cars[1].production_year = { 2018 };

drivers[9].license_number = 128456789;
drivers[9].name = "Hoda Malik";
drivers[9].birthdate = { 10, 2, 1980 };
drivers[9].cars[0].plate_number = 623;
drivers[9].cars[0].model = "Nissan Rogue";
drivers[9].cars[0].production_year = { 2015 };
drivers[9].cars[1].plate_number = 727;
drivers[9].cars[1].model = "Dodge Challenger";
drivers[9].cars[1].production_year = { 2016 };
drivers[9].cars[2].plate_number = 528;
drivers[9].cars[2].model = "BMW 230";
```

```
        drivers[9].cars[2].production_year = { 2017 };

    };

Update updates[3005];
void AssignFine() {
        cout << "Note: Maximum allowed road speed is 70 km/h.\n";
        hasFoundCar = false;
    int car_speed;
    string street_name;
        Date fine_assignment_date;
        cout << "Street name: ";
        cin.ignore();
        getline(cin, street_name);
        do {
                cout << "\nCar speed: ";
                cin >> car_speed;
                if (car_speed > 450)
                        cout << "Speed can't exceed 450 km/h.";
                else if (car_speed < 0)
                        cout << "Speed can't be a negative number.\n";
        } while (car_speed > 450 || car_speed < 0);
        cout << "\nCar plate number: ";
        cin >> plate_number;
        do {
                cout << "\nDate: Day) ";
                cin >> fine_assignment_date.day;
                if (fine_assignment_date.day <= 0 || fine_assignment_date.day > 31)
                        cout << "Day must be a number from 1 to 31.";
        } while (fine_assignment_date.day <= 0 || fine_assignment_date.day > 31);
        do {
                cout << "Month) ";
                cin >> fine_assignment_date.month;
                if (fine_assignment_date.month <= 0 || fine_assignment_date.month > 12)
                        cout << "Month must be a number from 1 to 12.\n";
        } while (fine_assignment_date.month <= 0 || fine_assignment_date.month > 12);
                cout << "Year) ";
                cin >> fine_assignment_date.year;
    if (car_speed <= kRoad_maximum_speed)
                cout << "No fine has been assigned, The Car speed didn't exceed maximum allowed
road speed.\n";
        else {
                for (int drivers_counter = 0; drivers_counter < kNumber_of_drivers;
drivers_counter++) {
                        for (int cars_counter = 0; cars_counter < kMax_cars_number;
cars_counter++) {
                                for (int fines_counter = 0; fines_counter < kMax_fines_number;
fines_counter++) {
                                        if
(drivers[drivers_counter].cars[cars_counter].plate_number == plate_number &&

        drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value == 0) {
                                                hasFoundCar = true;

        drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value = (car_speed -
kRoad_maximum_speed) * 2;

        drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date =
fine_assignment_date;

        drivers[drivers_counter].cars[cars_counter].fines[fines_counter].status = "Unpaid.";
```

```
       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].steet_name =
street_name;
                                          drivers[drivers_counter].total_unpaid += (car_speed
- kRoad_maximum_speed) * 2;
                                          cout << "Done !\n";
                                          return;
                                   }
                            }
                     }
          }
       if (!hasFoundCar)
              cout << "There is no car with this plate number.\n";
       }
}
void PayByPlate() {
       hasFoundFine = false;
       hasFoundCar = false;
       cout << "Please enter your plate number.\n";
    counter = 0;
       cin >> plate_number;
       for (int drivers_counter = 0; drivers_counter < kNumber_of_drivers; drivers_counter++)
{
              for (int cars_counter = 0; cars_counter < kMax_cars_number; cars_counter++) {
                if (drivers[drivers_counter].cars[cars_counter].plate_number == plate_number)
{
                     hasFoundCar = true;
                     for (int fines_counter = 0; fines_counter < kMax_fines_number;
fines_counter++) {
                            if
(drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value != 0 &&

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].status == "Unpaid.")
{
                                          hasFoundFine = true;
                              //to be displayed only once when the counter is equal to zero
                              if(!counter)
                                   cout << "\tList of fines\n";
                                   cout << counter + 1 << ") Value: " <<
drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value << "\nIn street: " <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].steet_name <<
"\nDate: Day)" <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.day << "\tMonth)
" <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.month <<
"\tyear) " <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.year << "\n";;
                                   updates[counter] = { drivers_counter, cars_counter,
fines_counter };
                                   counter++;
                            }
                     }
                 }
              }
       }
       if (!hasFoundCar)
              cout << "There is no car with this plate number.\n";
       else if (!hasFoundFine)
```

```
              cout << "There are no unpaid fines assigned to this car.\n";
       else {
              cout << "Enter the number of the fine you would like to pay.\n";
                  cin >> fine_choice;
            if(fine_choice <= counter && fine_choice != 0) {
                            cout << "Done !\n";
                            drivers[updates[fine_choice -
1].which_driver].cars[updates[fine_choice - 1].which_car].fines[updates[fine_choice -
1].which_fine].status = "Paid.";
                            drivers[updates[fine_choice - 1].which_driver].total_unpaid -=
drivers[updates[fine_choice - 1].which_driver].cars[updates[fine_choice - 1].which_car].
                            fines[updates[fine_choice - 1].which_fine].value;
                     }
                     else
                            cout << "Invalid choice !\n";
       }

    }

void PayByName() {
       string name;
       cout << "Please enter your name. Make sure you capitalize the first letter of each
name.\n";
       cin.ignore();
       getline(cin, name);
    counter = 0;
       hasFoundCar = false;
       hasFoundFine = false;
       for (int drivers_counter = 0; drivers_counter < kNumber_of_drivers; drivers_counter++)
{
              if (drivers[drivers_counter].name == name) {
                     hasFoundCar = true;
                for (int cars_counter = 0; cars_counter < kMax_cars_number; cars_counter++) {
                       for (int fines_counter = 0; fines_counter < kMax_fines_number;
fines_counter++) {

if(drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value != 0 &&

drivers[drivers_counter].cars[cars_counter].fines[fines_counter].status == "Unpaid.") {
                                     hasFoundFine = true;
                                     //to be displayed only once when the counter is equal to
zero
                                     if (!counter)
                                            cout << "\tList of fines\n";
                                     cout << counter + 1 << ")\nCar model: " <<
drivers[drivers_counter].cars[cars_counter].model << "\nValue: " <<

drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value << "\nIn street: " <<

drivers[drivers_counter].cars[cars_counter].fines[fines_counter].steet_name << "\nDate: Day)"
<<

drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.day << "\tMonth) " <<

drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.month << "\tyear) " <<

drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.year << "\n";
                                     updates[counter] = { drivers_counter, cars_counter,
fines_counter };
                                     counter++;
                              }
                       }
                }
```

```
			}
		}
		if (!hasFoundCar)
			cout << "There's no driver with this name.\n";
		else if (!hasFoundFine)
			cout << "There are no unpaid fines assigned to this driver.\n";
		else {
			cout << "Enter the number of the fine you would like to pay.\n";
			cin >> fine_choice;
			if (fine_choice <= counter && fine_choice != 0) {
				drivers[updates[fine_choice - 1].which_driver].cars[updates[fine_choice
- 1].which_car].fines[updates[fine_choice - 1].which_fine].status = "Paid.";
				drivers[updates[fine_choice - 1].which_driver].total_unpaid -=
				drivers[updates[fine_choice - 1].which_driver].cars[updates[fine_choice
- 1].which_car].fines[updates[fine_choice - 1].which_fine].value;
				cout << "Done !\n";
			}
			else
				cout << "Invalid choice !\n";
		}
}
void PayByLicense() {
		int license_number;
		counter = 0;
		cout << "Please enter your license number.\n";
		cin >> license_number;
	hasFoundCar = false;
		hasFoundFine = false;
		for (int drivers_counter = 0; drivers_counter < kNumber_of_drivers; drivers_counter++)
{
			if (drivers[drivers_counter].license_number == license_number) {
				hasFoundCar = true;
				for (int cars_counter = 0; cars_counter < kMax_cars_number;
cars_counter++) {
					for (int fines_counter = 0; fines_counter < kMax_fines_number;
fines_counter++) {
						if
(drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value != 0 &&

		drivers[drivers_counter].cars[cars_counter].fines[fines_counter].status == "Unpaid.")
{
							hasFoundFine = true;
							//to be displayed only once when the counter is
equal to zero
							if (!counter)
								cout << "\tList of fines\n";
							cout << counter + 1 << ")\nCar model: " <<
drivers[drivers_counter].cars[cars_counter].model << "\nValue: " <<

		drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value << "\nIn
street: " <<

		drivers[drivers_counter].cars[cars_counter].fines[fines_counter].steet_name <<
"\nDate: Day)" <<

		drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.day << "\tMonth)
" <<

		drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.month <<
"\tyear) " <<

		drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.year << "\n";
```

```
                                          updates[counter] = { drivers_counter, cars_counter,
fines_counter };
                                          counter++;
                                   }
                            }
                     }
              }
       }
       if (!hasFoundCar)
              cout << "There's no driver with this license number.\n";
       else if (!hasFoundFine)
              cout << "There are no unpaid fines assigned to this driver.\n";
       else {
              cout << "Enter the number of the fine you would like to pay.\n";
              cin >> fine_choice;
              if (fine_choice <= counter && fine_choice != 0) {
                     drivers[updates[fine_choice - 1].which_driver].cars[updates[fine_choice
- 1].which_car].fines[updates[fine_choice - 1].which_fine].status = "Paid.";
                     drivers[updates[fine_choice - 1].which_driver].total_unpaid -=
                     drivers[updates[fine_choice - 1].which_driver].cars[updates[fine_choice
- 1].which_car].fines[updates[fine_choice - 1].which_fine].value;
                     cout << "Done !\n";
              }
              else
                     cout << "Invalid choice !\n";
       }
}
void PayFine() {
       cout << "Would you like to choose a car using:\n1: Owner name ?\n2: Plate number ?\n3:
License number ?\n";
       int payment_method;
       cin >> payment_method;
       switch (payment_method) {
       case 1:
              PayByName();
              break;
       case 2:
              PayByPlate();
              break;
       case 3:
              PayByLicense();
              break;
       default:
              cout << "Invalid choice !\n";
       }
}
void SearchByName() {
       cout << "Please enter a name. Make sure you capitalize each new name.\n";
       string name;
       hasFoundCar = false;
       cin.ignore();
       getline(cin, name);
       for (int drivers_counter = 0; drivers_counter < kNumber_of_drivers; drivers_counter++)
{
              if (drivers[drivers_counter].name == name) {
                     hasFoundCar = true;
                     cout << "\tTotal unpaid fines: " <<
drivers[drivers_counter].total_unpaid << "\n";
                     cout << "Driver's birthday: " << drivers[drivers_counter].birthdate.day
<< '/' << drivers[drivers_counter].birthdate.month << '/' <<
                            drivers[drivers_counter].birthdate.year << ",\tDriver's license
number: " << drivers[drivers_counter].license_number << ".\n";
```

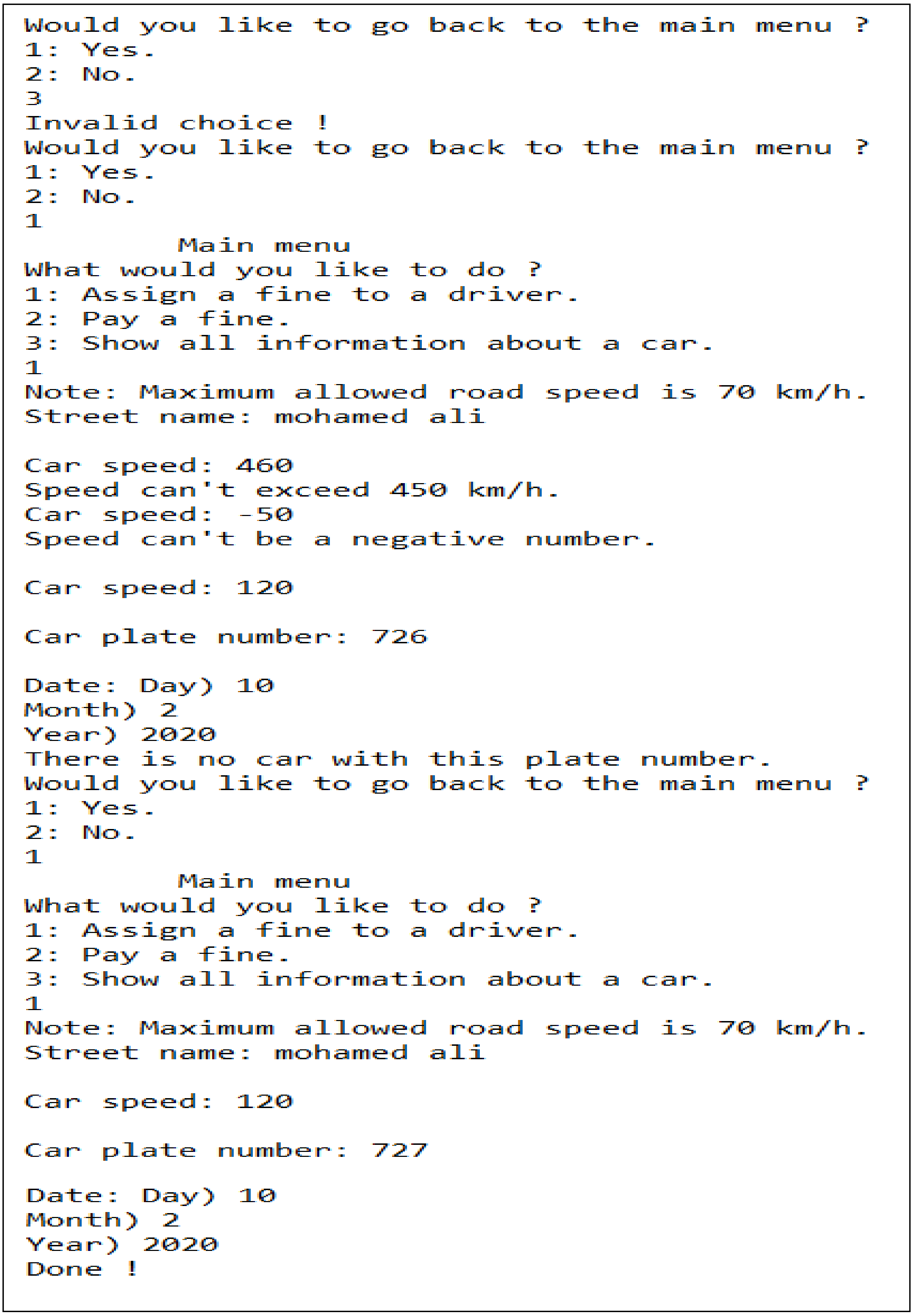
```
                    for (int cars_counter = 0; cars_counter < kMax_cars_number;
cars_counter++) {
                           //because Kmax_cars__number is 3 and the driver may own less
cars.
                           if (drivers[drivers_counter].cars[cars_counter].plate_number ==
0)
                                  break;
                           //to be displayed only once when the cars_counter is equal to
zero
                           if (!cars_counter)
                                  cout << "\tList of owned cars\n";
                           cout << cars_counter + 1 << ")\n";
                           cout << "Car model: " <<
drivers[drivers_counter].cars[cars_counter].model << ",\tCar plate number: "
<<drivers[drivers_counter].
                           cars[cars_counter].plate_number << ",\tCar production year: " <<
drivers[drivers_counter].cars[cars_counter].production_year << ".\n";
                               //there's no need to iterate over kMax fines_number if there
isn't any fine assigned
                                  if
(drivers[drivers_counter].cars[cars_counter].fines[0].value == 0) {
                                         cout << "There are no fines assigned to this
car.\n";
                                         continue;
                                  }
                           for (int fines_counter = 0; fines_counter < kMax_fines_number;
fines_counter++) {
                                  if
(drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value != 0) {
                                         //to be displayed only once when the fines_counter
is equal to zero
                                         if (!fines_counter)
                                                cout << "\tFines list\n";
                                             cout << fines_counter + 1 << ". Value: " <<
drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value <<
                                                ", \tStatus: " <<
drivers[drivers_counter].cars[cars_counter].fines[fines_counter].status << "\nIn street: " <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].steet_name <<
"\nDate: Day)" <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.day << "\tMonth)
" <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.month <<
"\tyear) " <<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.year << "\n";
                                  }
                           }
                    }
             }
       }
       if (!hasFoundCar)
             cout << "There is no driver with this name.\n";

   }

void SearchByPlateNumber() {
       cout << "Please enter a car plate number.\n";
       cin >> plate_number;
    hasFoundCar = false;
```

```
       for (int drivers_counter = 0; drivers_counter < kNumber_of_drivers; drivers_counter++)
{
                     for (int cars_counter = 0; cars_counter < kMax_cars_number;
cars_counter++) {
                            if (drivers[drivers_counter].cars[cars_counter].plate_number ==
plate_number) {
                          hasFoundCar = true;
                             cout << "Driver's name: " << drivers[drivers_counter].name <<
"\n";
                          cout << "\tTotal unpaid fines: " <<
drivers[drivers_counter].total_unpaid << "\n";
                          cout << "Driver's birthday: " <<
drivers[drivers_counter].birthdate.day << '/' << drivers[drivers_counter].birthdate.month <<
'/' <<
                             drivers[drivers_counter].birthdate.year << ",\tDriver's license
number: " << drivers[drivers_counter].license_number << ".\n";
                             cout << "Car model: " <<
drivers[drivers_counter].cars[cars_counter].model << ",\tCar plate number: "
<<drivers[drivers_counter].
                            cars[cars_counter].plate_number << ",\tCar production year: " <<
drivers[drivers_counter].cars[cars_counter].production_year << ".\n";
                            //there's no need to iterate over kMax fines_number if there
isn't any fine assigned
                            if (drivers[drivers_counter].cars[cars_counter].fines[0].value ==
0)
                                  cout << "There are no fines assigned to this car.\n";
                            else {
                               for (int fines_counter = 0; fines_counter < kMax_fines_number;
fines_counter++) {
                                     if
(drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value != 0) {
                                          //to be displayed only once when the fines_counter
is equal to zero
                                          if (!fines_counter)
                                                 cout << "\tFines list\n";
                                              cout << fines_counter + 1 << ". Value: " <<
drivers[drivers_counter].cars[cars_counter].fines[fines_counter].value <<
                                                 ",\tStatus: " <<
drivers[drivers_counter].cars[cars_counter].fines[fines_counter].status << "\nIn street: "<<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].steet_name<<"\nDate:
Day)"<<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.day<<"\tMonth)
"<<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.month <<
"\tyear) "<<

       drivers[drivers_counter].cars[cars_counter].fines[fines_counter].date.year<<"\n";
                                     }
                                }
                               }
                            }
                     }
       }
       if (!hasFoundCar)
              cout << "There is no car with this plate number.\n";
}
void ShowInformation() {
       int search_choice;
       cout << "You can search via:\n1: Owner name.\n2: Plate number.\n";
```

```
	cin >> search_choice;
	switch (search_choice) {
	case 1:
		SearchByName();
		break;
	case 2:
		  SearchByPlateNumber();
		break;
	default:
		cout << "Invalid choice !\n";
	}
}
int main() {
	system("color F0");
	SystemData();
	int use_again, service_choice;
	do {
		cout << "\tMain menu\nWhat would you like to do ?\n";
		cout << "1: Assign a fine to a driver.\n2: Pay a fine.\n3: Show all information
about a car.\n";
		cin >> service_choice;
		switch (service_choice) {
		case 1:
			AssignFine();
			break;
		case 2:
			PayFine();
			break;
		case 3:
			ShowInformation();
			break;
		default:
			cout << "Invalid choice !\n";
		}
		do {
			cout << "Would you like to go back to the main menu ?\n1: Yes.\n2:
No.\n";
			cin >> use_again;
			if (use_again != 1 && use_again != 2)
				cout << "Invalid choice !\n";
		} while (use_again != 1 && use_again != 2);
	} while (use_again == 1);
	return 0;
   }
```

6. Testing

```
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
4
Invalid choice !
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
1
Please enter your name. Make sure you capitalize the first letter of each name.
Hoda Malik
There are no unpaid fines assigned to this driver.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
1
Note: Maximum allowed road speed is 70 km/h.
Street name: abbas el akkad

Car speed: 450

Car plate number: 623

Date: Day) 50
Day must be a number from 1 to 31.
Date: Day) 12
Month) 15
Month must be a number from 1 to 12.
Month) 9
Year) 2020
Done !
```



```
Would you like to go back to the main menu ?
1: Yes.
2: No.
3
Invalid choice !
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
1
Note: Maximum allowed road speed is 70 km/h.
Street name: mohamed ali

Car speed: 460
Speed can't exceed 450 km/h.
Car speed: -50
Speed can't be a negative number.

Car speed: 120

Car plate number: 726

Date: Day) 10
Month) 2
Year) 2020
There is no car with this plate number.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
1
Note: Maximum allowed road speed is 70 km/h.
Street name: mohamed ali

Car speed: 120

Car plate number: 727

Date: Day) 10
Month) 2
Year) 2020
Done !
```

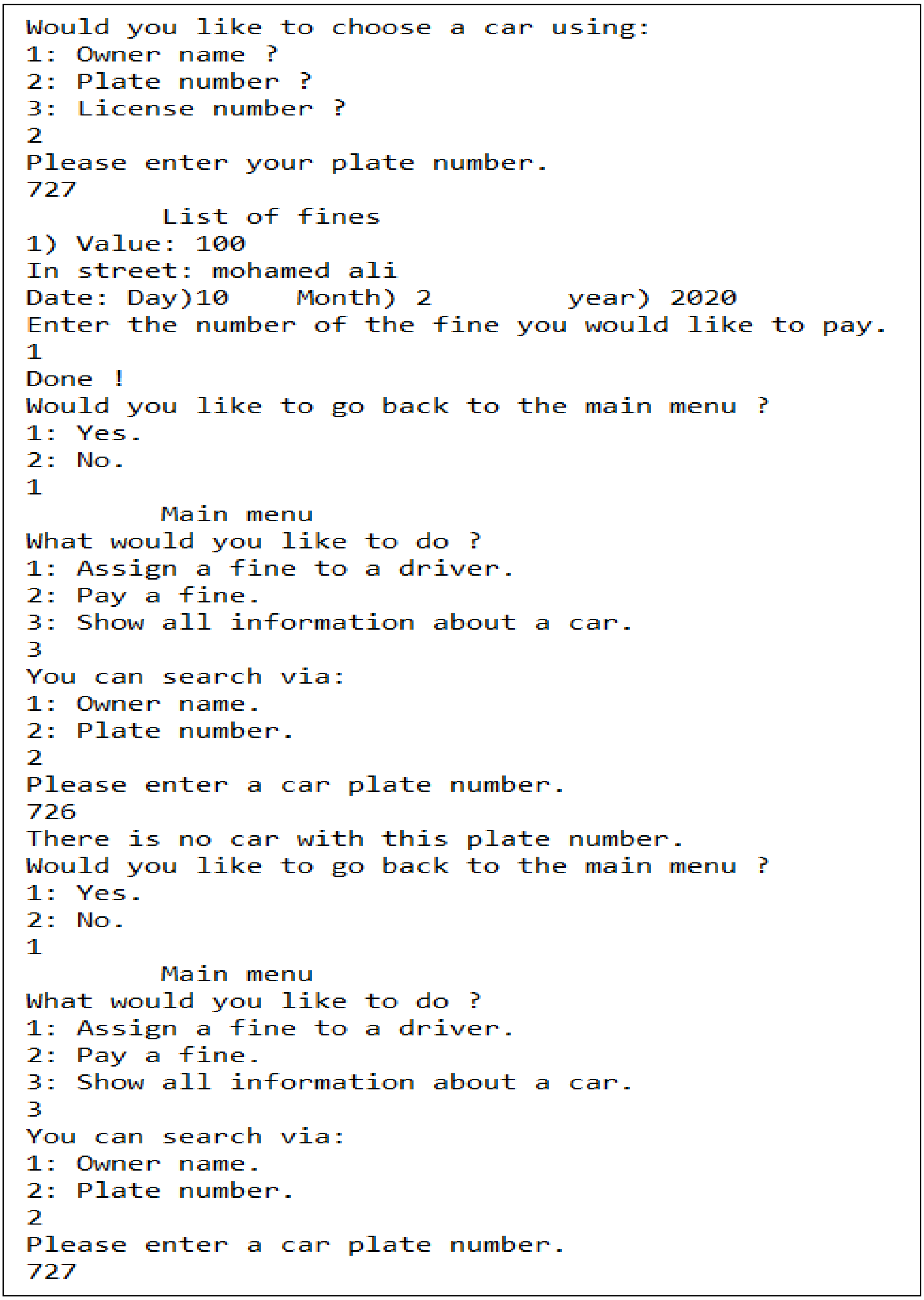
```
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
1
Note: Maximum allowed road speed is 70 km/h.
Street name: ramsis

Car speed: 50

Car plate number: 528

Date: Day) 6
Month) 5
Year) 2019
No fine has been assigned, The Car speed didn't exceed maximum allowed road speed.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
3
You can search via:
1: Owner name.
2: Plate number.
3
Invalid choice !
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
3
```

```
You can search via:
1: Owner name.
2: Plate number.
1
Please enter a name. Make sure you capitalize each new name.
Hoda Malik
        Total unpaid fines: 860
Driver's birthday: 10/2/1980,   Driver's license number: 128456789.
        List of owned cars
1)
Car model: Nissan Rogue,        Car plate number: 623,  Car production year: 2015.
        Fines list
1. Value: 760,  Status: Unpaid.
In street: abbas el akkad
Date: Day)12    Month) 9        year) 2020
2)
Car model: Dodge Challenger,    Car plate number: 727,  Car production year: 2016.
        Fines list
1. Value: 100,  Status: Unpaid.
In street: mohamed ali
Date: Day)10    Month) 2        year) 2020
3)
Car model: BMW 230,     Car plate number: 528,  Car production year: 2017.
There are no fines assigned to this car.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
1
Please enter your name. Make sure you capitalize the first letter of each name.
Hoda Ahmed
There's no driver with this name.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
```

```
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
1
Please enter your name. Make sure you capitalize the first letter of each name.
Hoda Malik
        List of fines
1)
Car model: Nissan Rogue
Value: 760
In street: abbas el akkad
Date: Day)12    Month) 9        year) 2020
2)
Car model: Dodge Challenger
Value: 100
In street: mohamed ali
Date: Day)10    Month) 2        year) 2020
Enter the number of the fine you would like to pay.
3
Invalid choice !
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
1
Please enter your name. Make sure you capitalize the first letter of each name.
Hoda Malik
```

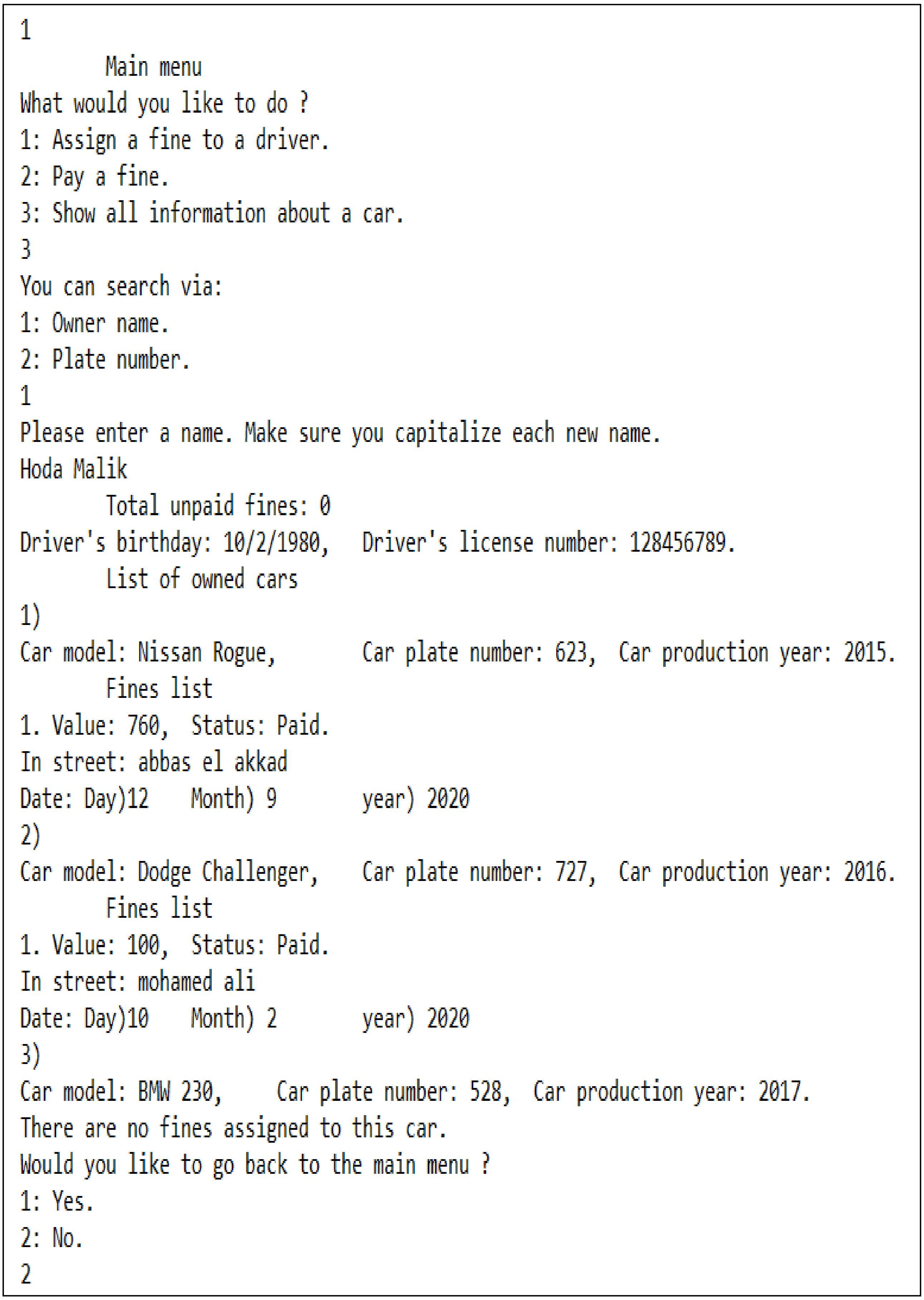
```
        List of fines
1)
Car model: Nissan Rogue
Value: 760
In street: abbas el akkad
Date: Day)12    Month) 9        year) 2020
2)
Car model: Dodge Challenger
Value: 100
In street: mohamed ali
Date: Day)10    Month) 2        year) 2020
Enter the number of the fine you would like to pay.
1
Done !
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
3
You can search via:
1: Owner name.
2: Plate number.
1
Please enter a name. Make sure you capitalize each new name.
hoda malik
There is no driver with this name.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
3
You can search via:
1: Owner name.
2: Plate number.
1
```

```
Please enter a name. Make sure you capitalize each new name.
Hoda Malik
        Total unpaid fines: 100
Driver's birthday: 10/2/1980,   Driver's license number: 128456789.
        List of owned cars
1)
Car model: Nissan Rogue,        Car plate number: 623,  Car production year: 2015.
        Fines list
1. Value: 760,  Status: Paid.
In street: abbas el akkad
Date: Day)12    Month) 9        year) 2020
2)
Car model: Dodge Challenger,    Car plate number: 727,  Car production year: 2016.
        Fines list
1. Value: 100,  Status: Unpaid.
In street: mohamed ali
Date: Day)10    Month) 2        year) 2020
3)
Car model: BMW 230,     Car plate number: 528,  Car production year: 2017.
There are no fines assigned to this car.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
2
Please enter your plate number.
726
There is no car with this plate number.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
```



```
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
2
Please enter your plate number.
727
        List of fines
1) Value: 100
In street: mohamed ali
Date: Day)10    Month) 2        year) 2020
Enter the number of the fine you would like to pay.
1
Done !
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
3
You can search via:
1: Owner name.
2: Plate number.
2
Please enter a car plate number.
726
There is no car with this plate number.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
3
You can search via:
1: Owner name.
2: Plate number.
2
Please enter a car plate number.
727
```

```
Driver's name: Hoda Malik
        Total unpaid fines: 0
Driver's birthday: 10/2/1980,   Driver's license number: 128456789.
Car model: Dodge Challenger,    Car plate number: 727,  Car production year: 2016.
        Fines list
1. Value: 100,  Status: Paid.
In street: mohamed ali
Date: Day)10    Month) 2        year) 2020
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
3
Please enter your license number.
128456788
There's no driver with this license number.
Would you like to go back to the main menu ?
1: Yes.
2: No.
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
2
Would you like to choose a car using:
1: Owner name ?
2: Plate number ?
3: License number ?
3
Please enter your license number.
128456789
There are no unpaid fines assigned to this driver.
Would you like to go back to the main menu ?
1: Yes.
2: No.
```



```
1
        Main menu
What would you like to do ?
1: Assign a fine to a driver.
2: Pay a fine.
3: Show all information about a car.
3
You can search via:
1: Owner name.
2: Plate number.
1
Please enter a name. Make sure you capitalize each new name.
Hoda Malik
        Total unpaid fines: 0
Driver's birthday: 10/2/1980,   Driver's license number: 128456789.
        List of owned cars
1)
Car model: Nissan Rogue,        Car plate number: 623,  Car production year: 2015.
        Fines list
1. Value: 760,  Status: Paid.
In street: abbas el akkad
Date: Day)12    Month) 9        year) 2020
2)
Car model: Dodge Challenger,    Car plate number: 727,  Car production year: 2016.
        Fines list
1. Value: 100,  Status: Paid.
In street: mohamed ali
Date: Day)10    Month) 2        year) 2020
3)
Car model: BMW 230,     Car plate number: 528,  Car production year: 2017.
There are no fines assigned to this car.
Would you like to go back to the main menu ?
1: Yes.
2: No.
2
```

References:

[1] http://www.cs.technion.ac.il/users/yechiel/c++-faq/scope-of-for-loop-_var.html

[2] https://users.ece.cmu.edu/~eno/coding/CppCodingStandard.html#linele

[3] http://umich.edu/~eecs381/handouts/C++_Coding_Standards.pdf

[4] http://llvm.org/docs/CodingStandards.html#include-as-little-as-possible

[5] https://www.samanthaming.com/tidbits/34-better-boolean-variable-names/