Assignment 4
Due: Friday 11:59pm on Nov 8, 2024

Group Members:

Member #1:

First Name: Eric
Last Name: Morse
Student ID: 1141504

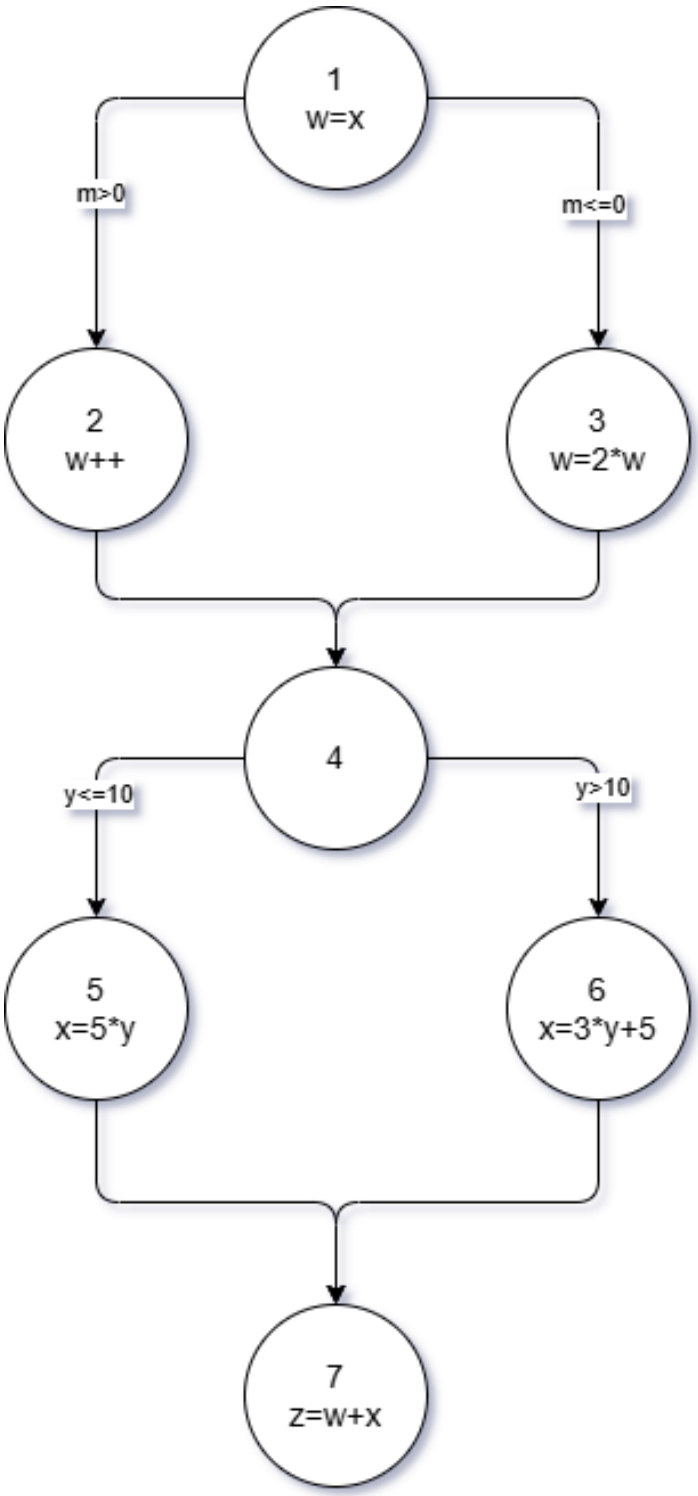
Member #2:

First Name: Or
Last Name: Brener
Student ID: 1140102

Member #3:

First Name: Danindu
Last Name: Marasinghe
Student ID: 1093791

1. Chapter 7.3, Question 1 (a) [2 marks]



2. Chapter 7.3, Question 2 [10 marks]

Select a commercial coverage tool of your choice. Note that some have free trial evaluations. Choose a tool, download it, and run it on some software. You can use one of the examples from this text, software from your work environment, or software available over the Web. Write up a short summary report of your experience with the tool. The main grading criterion is that you actually collect some coverage data for a reasonable set of tests on some program." You may use any programming language you wish; I would recommend Java, to be consistent with the textbook, however, Python, C, C#, C++, and others would be just fine. Provide your report, 1-2 pages in the space below.

Experience with Python and Pytest

Pytest offers a simple, intuitive syntax for writing test cases in Python. My experience with it has been positive due to its rich set of features, including easy-to-read assert statements, parameterization options, and plugins that extend functionality. The installation process is straightforward, requiring only a `pip install pytest` command, and Pytest automatically detects test files and functions, which reduces the manual setup usually associated with testing frameworks.

Code Coverage Measurement

Code coverage is a crucial metric in assessing the completeness of tests, and Pytest integrates well with the **Coverage.py** tool to measure it. With the addition of the `pytest-cov` plugin (`pip install pytest-cov`), I was able to generate a code coverage report, which includes the percentage of code lines covered by tests and a detailed summary of which lines were executed. Running `pytest --cov=your_module_name` gives a clear breakdown of how well the codebase is covered by tests.

One thing I learned is that high code coverage doesn't always guarantee perfect tests but rather serves as a valuable indicator of areas that may require more comprehensive testing. The coverage reports generated by Pytest-Cov allow me to focus on untested branches and functions and ensure that critical code paths are tested.

Example Code File and Test Cases

Example Code: `calculator.py`

A4 > Q2 > calculator.py > divide

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b

```

Example Test Cases: `test_calculator.py`

A4 > Q2 > test_calculator.py > ...

```python
import pytest
from calculator import add, subtract, multiply, divide

def test_add():
    assert add(3, 5) == 8
    assert add(-1, 1) == 0

def test_subtract():
    assert subtract(10, 5) == 5
    assert subtract(0, 3) == -3

def test_multiply():
    assert multiply(4, 3) == 12
    assert multiply(-1, 5) == -5

def test_divide():
    assert divide(10, 2) == 5
    assert divide(9, 3) == 3
    with pytest.raises(ValueError):
        divide(5, 0)

```

Coverage Data:

```
test_calculator.py ....                                                  [100%]

---------- coverage: platform win32, python 3.11.1-final-0 -----------
Name                Stmts   Miss  Cover
---------------------------------------
calculator.py          10      0   100%
test_calculator.py     16      0   100%
---------------------------------------
TOTAL                  26      0   100%


============================================== 4 passed in 0.11s ===============================================
PS C:\Users\danin\My Drive\School\UNI\Year 6\Testing\TestingCode\CIS4150\A4\Q2>
```

3. Chapter 7.6, Question 1 [19 Marks]

a. Construct the Get Balance use case and use case scenarios for interactions with a bank Automated Teller Machine. Include the following [12 marks in total]

Use Case Name: Get Balance
Summary [0.5 marks]: Customer uses a valid card to check their current funds from a valid bank account.
Actor [0.5 marks]: ATM, Customer
Precondition [0.5 marks]: ATM is displaying the idle welcome message
Description [4 marks]:

1. Customers interest an ATM Card into the ATM Card Reader.
2. If the system can recognize the card, it reads the card number.
3. System prompts the customer for a PIN.
4. Customer enters PIN.
5. System checks the expiration date and whether the card has been stolen or lost.
6. If the card is valid, the system checks whether the PIN entered matches the card PIN.
7. If the PINs match, the system finds out what accounts the card can access.
8. System displays customer accounts and prompts the customer to choose a type of transaction. Three types of transactions are Withdraw Funds, Get Balance, and Transfer Funds, The previous eight steps are part of all three use cases; the following steps are unique to the Get Balance use case.
9. Customer selects Get Balances, selects account number, and enters the amount.
10. System checks that the account is valid
11. System displays the current account balance to the customer, as well as a continue button.
12. Customer clicks continue.
13. System displays customer accounts and prompts the customer to choose a type of transaction. Three types of transactions are Withdraw Funds, Get Balance, and Transfer Funds, and return.
14. Customer clicks return.
15. System ejects card.
16. System displays the idle welcome message.

Alternatives [4 marks]:

- If the system cannot recognize the card, it is ejected and a welcome message is displayed.
- If the current date is past the card's expiration date, the card is confiscated and a welcome message is displayed.
- If the card has been reported lost or stolen, it is confiscated and a welcome message is displayed.
- If the customer entered PIN does not match the PIN for the card, the system prompts for a new PIN.

- If the customer enters an incorrect PIN three times, the card is confiscated and a welcome message is displayed.
- If the account number entered by the user is invalid, the system displays an error message, ejects the card, and a welcome message is displayed.
- If the customer enters Cancel, the system cancels the transaction, ejects the card, and a welcome message is displayed.

Postcondition [0.5 marks]: Account Balance has been displayed to the Customer

One use case scenario for this use case [2 marks]: The customer inserts their ATM card, and the system recognizes it, prompting the customer for a PIN. The customer enters the PIN, and the system verifies that the card is valid (not expired, lost, or stolen). After the PIN is confirmed, the system displays the customer's accessible accounts and asks the customer to choose a transaction. The customer selects **Get Balance**, chooses the account, and the system verifies the account. It then displays the current balance for the chosen account along with a "Continue" option. The customer clicks "Continue," returning to the transaction menu, and then selects "Return." The system ejects the card and displays the idle welcome message.

b. Describe the Test Case for this use case. [7 marks]

Purpose: [0.5 marks]	To determine functionality of Get Balance
Prerequisite: [2 marks]	Working ATM that is on the idle screen, a valid ATM Card and PIN.
Test Data: [0.5 marks]	Valid ATM Card, PIN, and Expiry Date. Valid account with the balance $8472.12 linked to the ATM Card.
Steps: [2 marks]	1. Insert Valid Card 2. Insert PIN 3. Press Get Balance 4. Press out account number 5. Press continue 6. Press Return
Expected Output: [2 marks]	In the order we expect: 1. Pin enter screen 2. An option screen for Withdraw Funds, Get Balance, and Transfer Funds. 3. Account selection screen 4. The funds in the account, which should be $8472.12

	5. The option screen 6. The idle screen.

4. [BONUS] Building a Simple Traffic FSM [Bonus 4 marks]

Objective

Learn to use MATLAB Stateflow to create a basic Finite State Machine (FSM) for a simple traffic light system. The FSM will cycle through Red, Green, and Yellow states, simulating the operation of a traffic light.

Software Requirements

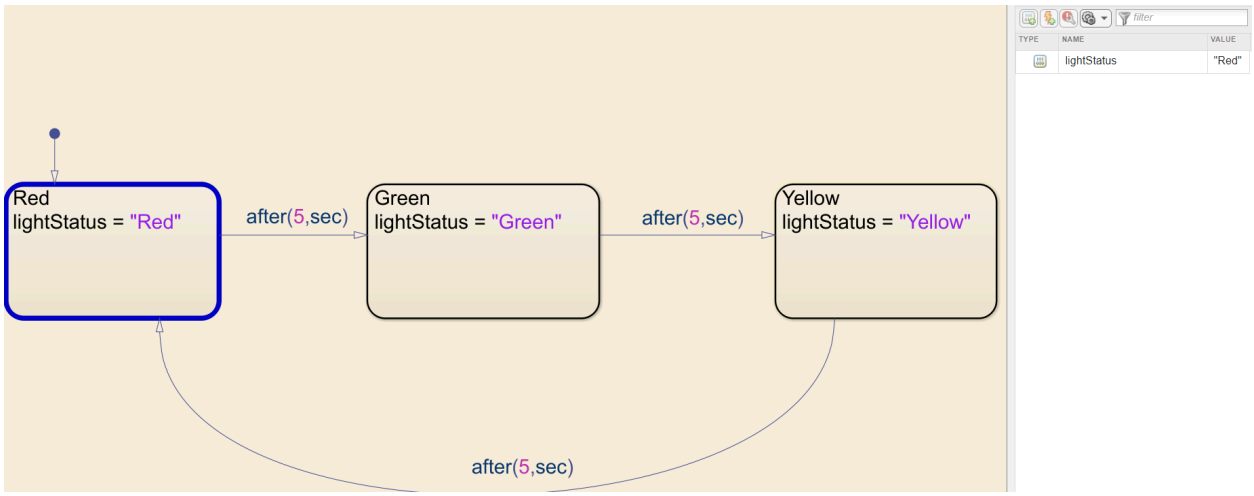
- MATLAB with the Stateflow add-on [download the 30-day free trial]:

Background

Stateflow in MATLAB is a graphical tool for modeling and simulating state machines and logic in dynamic systems. In this assignment, you will build an FSM representing a traffic light that changes states in response to time.

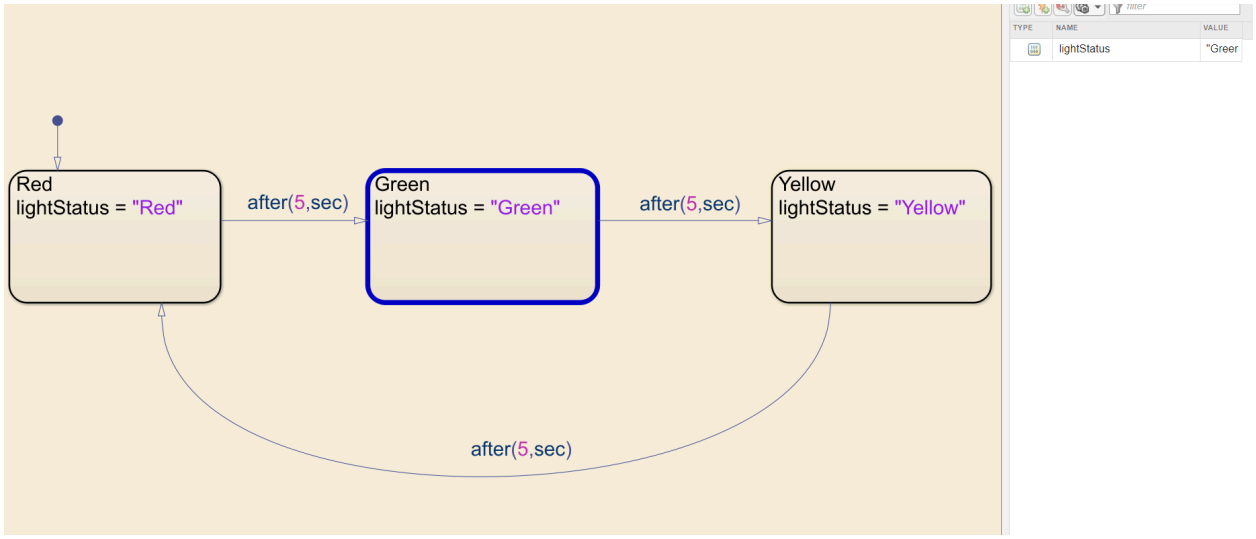
Rubric:

1. Correct State Setup [1 Mark]: Each state (Red, Green, Yellow) is set up correctly.
2. Transitions and Timing Conditions [1 Mark]: Transitions are created between each state with the correct timing conditions.
3. Output Signal [1 Mark]: The output variable lightStatus is correctly configured and reflects the traffic light state. You can also use the Lamp UI component to visually show a traffic light (red, yellow, green).
4. Simulation Results [1 Mark]: Simulation runs correctly, with observed transitions matching the expected cycle.
5. Documentation and Screenshot: Include clear and complete screenshots and descriptions in the space below.
6. Submit your MATLAB source code to CourseLink

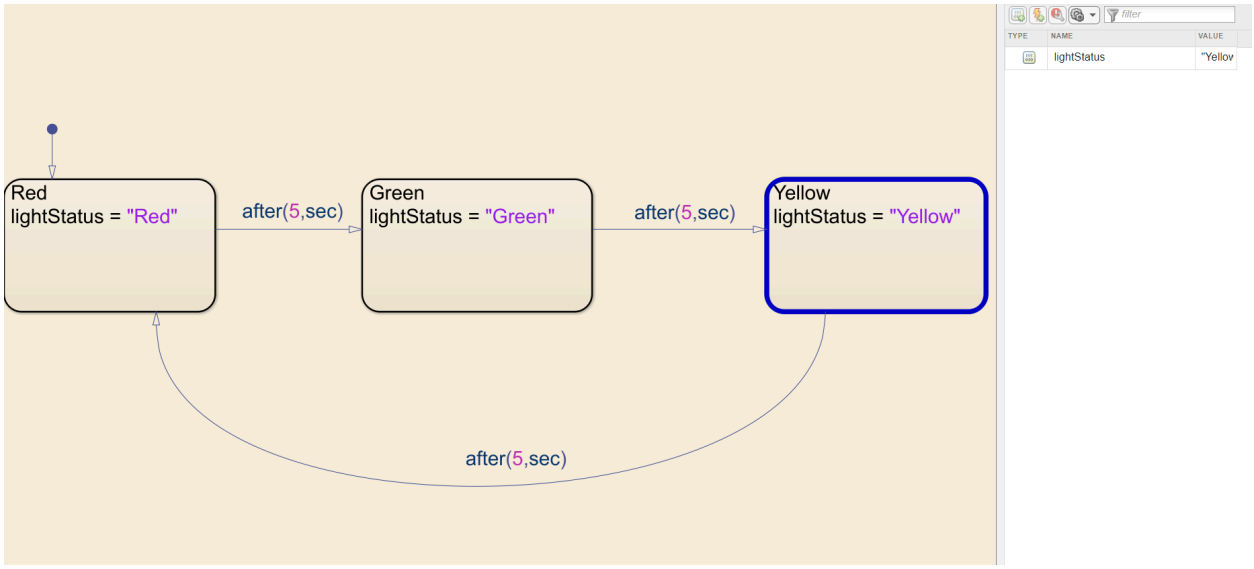


State: lightStatus = “Red”

The light is Red and is waiting to turn Green



State: lightStatus = “Green”
The light is green and it is waiting to turn yellow.



State: lightStatus = “Yellow”
The light is yellow and it is waiting to turn Red.