Workshop Grading and Promotion Policy

Workshops for this course will be assessed using the following criteria:

- Workshops are graded based on two components:
  1. Individual Logic Assignment (40%)
     - Individual work is due **2 days after the assigned date** (class) by **end of day 23:59 EST**
     - Individual logic assignments are to be done individually
     - Members who do not submit work on-time, will receive a zero grade for the workshop
     - Members who receive a zero grade for the individual part, will not be eligible to receive grades for the group solution part
  2. Sub-Group Overall Solution (60%)
     - Group solution is due **4 days after the assigned date** (class) by **end of day 23:59 EST**
     - **Name and ID of all contributing members must be stated at the top of all file submissions**
     - If not submitted on-time, a zero grade will be applied for the group portion of the workshop
     - If the submitted solution is essentially a copy of the individual parts thrown together containing no effort to properly integrate as a seamless overall solution, a zero grade will be applied for the group portion of the workshop

- A zero grade on a workshop will not be counted towards the minimum necessary number of completed workshops

- Video presentations are due **1 day after your next class** by **end of day 23:59 EST**
  - Each student must do a video presentation **at least once** by the end of the term and should minimally consist of the following:
    - Description of the problem and its solution in non-technical terms. You should assume your audience is non-technical and interested in using your application solution.

    - Market your application solution by providing sample screenshots of how you envision your application to look which should include a sample workflow demonstrating how easy it is to use

- You must **successfully complete 9 workshops** (if > 9 are completed, the best 9 will be used)
- Workshop solutions and presentations will be evaluated using the published workshop rubrics

Group Breakdown

Each group has **two sub-groups** determined by the assigned **member number**:

Sub-Group 1: **Members 1-3**

- **Member-1**: Responsible for doing workshop **Logic 1**
- **Member-2**: Responsible for doing workshop **Logic 2**
- **Member-3**: Responsible for doing workshop **Logic 3**

Sub-Group 2: **Members 4-6**

- **Member-4**: Responsible for doing workshop **Logic 1**
- **Member-5**: Responsible for doing workshop **Logic 2**
- **Member-6**: Responsible for doing workshop **Logic 3**

Sub-Group Solution

- Each sub-group is a team and **must work together** creating the overall group solution
- The group solution is not to be done by an individual. The group solution is expected to be a seamless solution (looking as though one person has done it) and has undergone refinement and testing to ensure the logic properly addresses the workshop problem.
- If the submitted work amounts to essentially copying and pasting everyone's logic part together, a zero grade will be applied for the group work portion.

Work Submission

All work must be emailed to your instructor. You must follow the email guidelines described below.

- All work submitted (applied to both individual and group submissions) **requires all contributing members names to be stated at the top of all files being submitted**

Email Subject Line

- Highlighted parts indicate your specific information
- There are no spaces
- **`APS145-[SECTION]-WS[#]:Group[#]`**
  - Example: **`APS145-NAA-WS1:Group3`**

File Attachment

Individual Work Submissions

Attach a file containing your work (**pseudo code** OR **flowchart**)

- Highlighted parts indicate your specific information
- **Pseudo code**: `logic[#].fullname.pseudocode.txt`
  - Example: `logic2.Cameron Gray.pseudocode.txt`
- **Flowchart**: `logic[#].fullname.flowchart.jpg` *(Note: .jpg or .png)*
  - Example: `logic3.Cameron Gray.flowchart.png`

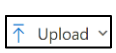
Sub-Group Solution Submission

Attach a file containing your group work (**pseudo code** OR **flowchart**)

- Highlighted parts indicate your specific information
- There are no spaces
- **Pseudo code**: `ws[#].group.pseudocode.txt`
  - Example: `ws1.group.pseudocode.txt`
- **Flowchart**: `ws[#].group.flowchart.jpg` *(Note: .jpg or .png)*
  - Example: `ws3.group.flowchart.png`

Presentation Submission

Video files can be quite large and will most likely be rejected by Seneca's email services. Therefore, you will have to **SHARE** your video file using your Seneca account Microsoft **ONE drive**.

- **Video file name**: `WS[#].fullname.video.mp4`
  - Example: `WS4.Cameron Gray.video.mp4`
- Go to https://myseneca.ca, click on (top left corner)  and select the One Drive application option
- Upload your video file: 

- Share  the file with your instructor: **Copy the shared link** 

- Paste the shared link into your email

Workshop - 3

Workshop Value: 10 marks (5% of your final grade)

Learning Outcomes

Upon successful completion of this workshop, you will have demonstrated the abilities:

- to decipher and identify a problem
- to analyze and decompose a problem
- to identify the required detailed steps to solve a problem
- to communicate the solution to fellow peers and non-technical business persons Workshop

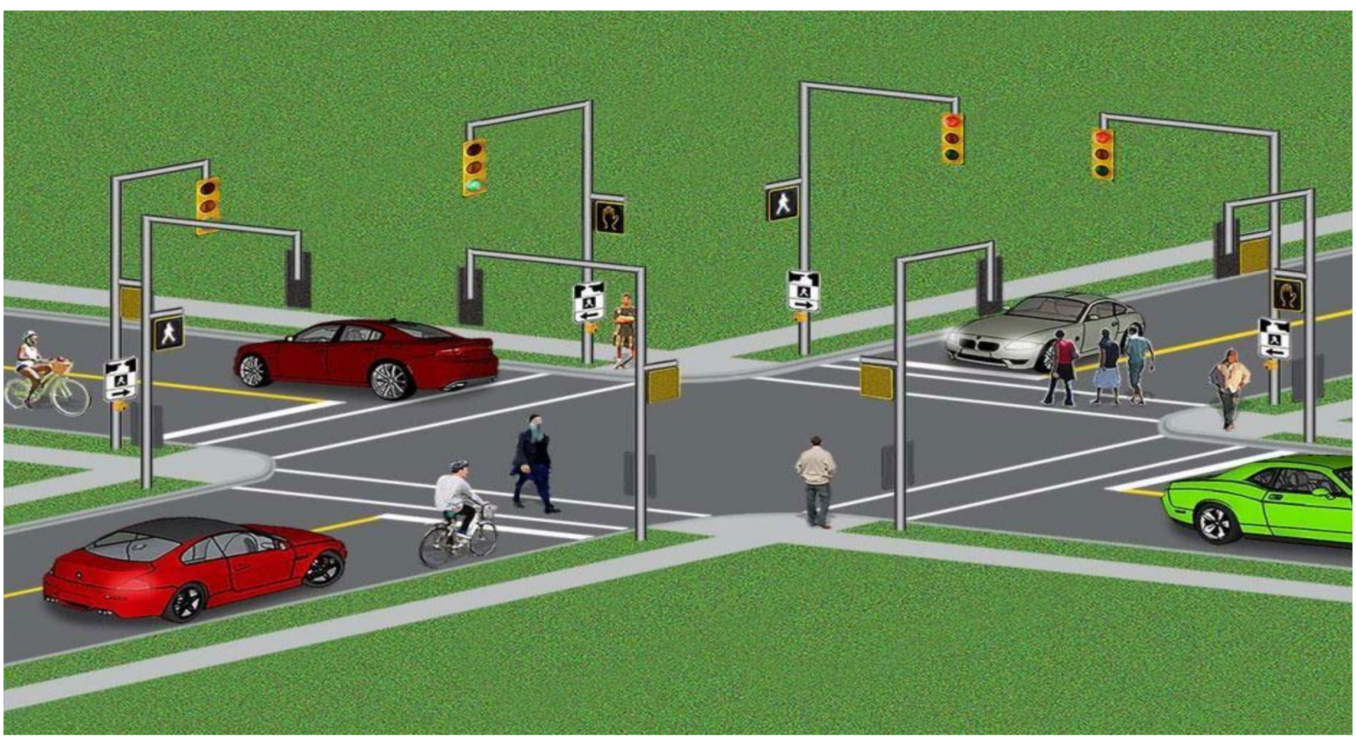
Workshop Overview

Finite state machines are around us everywhere. State machines put simply, are logical units that are limited to being in only one state at any given time. A finite state machine is one that is limited to a defined set of states (there are defined limits to the number of possible states it can be in).

Traffic lights are a good example of finite state machines. In a typical traffic light, there are 3 possible states: Green (go), Amber (don't begin to cross the intersection because the red light is about to trigger), and Red (stop). Each state is strictly timed (can be different in duration for each state) and synchronized with the opposing set of lights.

Workshop Details

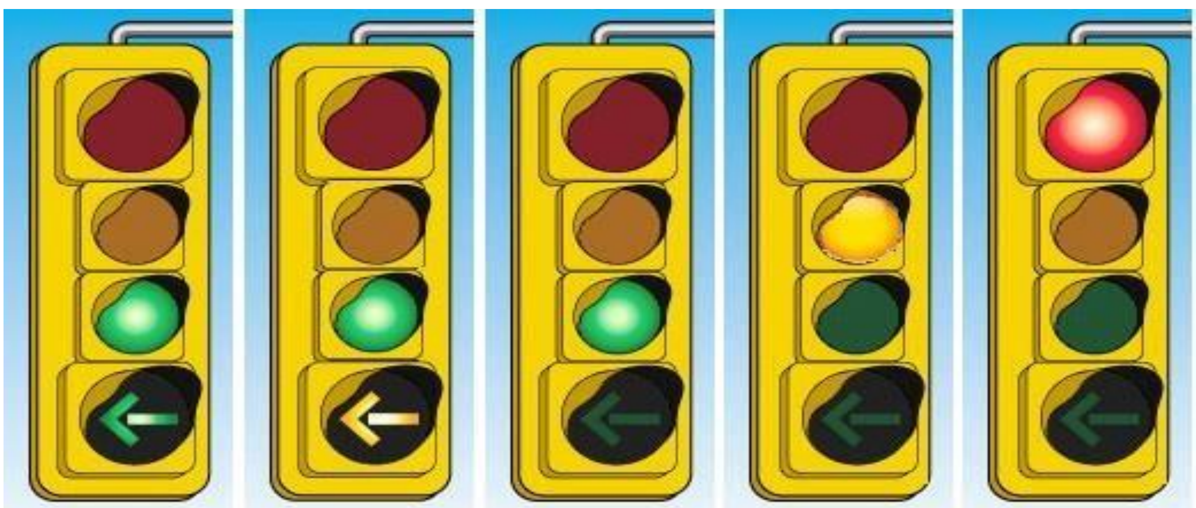
A busy intersection in downtown Toronto needs to upgrade the current 3-light 3-state traffic light system (see illustration below) to a **4-light 5-state** system to address an increasingly heavy demand for left turns. A **computer simulation** needs to be developed as a proof of concept.



Looking at the light that is red, we see the cars are stopped, waiting for the light to turn green. If 6 of more cars line up, the car sensor will signal the light to turn green sooner, as described below. The pedestrian who wants to go in the same direction as the cars stopped on the red light is also stopped. If the pedestrian presses the walk button on the light pole, this will also decrease the wait time for the red light as described below.

The 4 lights in the new system are as follows:

1. Red 
2. Amber 
3. Green 
4. Advance left turn (Green to Amber) 



4-Light 5-State Traffic Light: ***Sequence of states***

The sequence of a 4-light **5-state** system is shown in the above illustration. Below are the 5-states including the time duration for each state:

1. Solid Green + Advanced Left **Green** Arrow **[20 seconds]**
2. Solid Green + Advanced Left **Amber** Arrow **[5 seconds]**
3. Solid Green + Advanced Left Arrow (**off**) **[50 seconds]**
4. Solid **Amber** **[10 seconds]**
5. Solid **Red** **[?? seconds YOU FIGURE THIS OUT]**

Additional Features

Request-To-Walk Button

A button is provided for pedestrians to press to alert the system a person needs to cross. This is only effective when the lights are in a red-light state. This request will reduce the remaining wait time of the red-light state to 20 seconds.

Car Sensor

There is a sensor mounted on the light fixtures that monitors the number of waiting cars when in a red-light state. When the number of cars waiting reaches 6 or more, the sensor will trigger the system to reduce the remaining wait time of the red-light state to 20 seconds.

Interrupt

The defined system should take in to account an "interrupt". This is something that can occur at any time and will take over/supersede the active state. Examples of this type of interruption can be the traffic controllers setting all lights to flash amber (for caution) or to flash red (4-way stop) etc... **You don't need to specify what the interrupt should do – only account for it by checking if there is an interrupt to the normal process flow and how the system should resume when the interrupt has completed.**

Work Breakdown

[Logic 1] Define a standard 3-light system (red, green, amber) with no advance green, request to walk light, car censor, or interrupt.

[Logic 2] Define a red-light waiting sequence with a request to walk button and a car sensor.

[Logic 3] Define a standard 3-light system (red, green, amber) with an interrupt (no advance green, request to walk, or car sensor).

[Group] Define the 4-light 5-state system that supports the advanced left turn light, request to walk, car sensor, and interrupt features.

Your Tasks

1. Where applicable, apply the core components of the **computational thinking** approach to problem solving to help you synthesize a solution
2. Communicate the independent logic parts and group solutions using pseudo code/flowchart (see assignments below)
3. Create a video presentation to market your envisioned simulation application

Individual and sub-group assignments

Sub-Group 1 (pseudo code)				
Task	**Subtask**	**Member(s)**	**Marks**	**Comments**
Pseudocode	Logic 1	**1**	40%	Members are graded individually
	Logic 2	**2**	40%	
	Logic 3	**3**	40%	
	Group Solution	**1-3**	60%	Eligible members get same mark
Sub-Group 2 (flowchart)				
Task	**Subtask**	**Member(s)**	**Marks**	**Comments**
FlowChart	Logic 1	**4**	40%	Members are graded individually
	Logic 2	**5**	40%	
	Logic 3	**6**	40%	
	Group Solution	**4-6**	60%	Eligible members get same mark
Video	Presentation	**3 or 6**	100%	Members rotate weekly

* **Presentation**: Decide among yourselves which member among the entire group will be doing the presentation. Priority should be given to those who have not yet done one.