Coursework 1 - TOUCAN Crossing

You are required to implement a simulation of a simplified version of the TOUCAN crossing at the entrance to the King's Buildings at the junction of West Mains Road, Westfeild Road and Esslemount Road. Your simulation must display the signal lights on a blinkstick square. Two LED's are needed to display the trafic lights as both the Red and Amber lights must show on the same signal.

You must use the following LEDs

LED		Light
0	Esslemount Road	Stop
1	Esslemount Road	Caution / Go
2	Crossing	Go
3	Crossing	Danger
4	West Mains Road	Caution / Go
5	West Mains Road	Stop
6	Mayfeild Road	Caution / Go
7	Mayfeild Road	Stop

Timings must follow section 18 of Chapter 6 "Traffic Control" of the UK Traffic Signs Manual, 2019. Please remember that the sequence of traffic lights for vehicles in the UK is red (Stop), red & amber (prepare), green (go), amber (stop is safe to do so), red (stop).

An important simplifcation is that no filters are to be implemented. Traffic on Mayfeild Road is stopped in both directions at the same time (i.e. there is no turn left filter). Traffic on Esslemount road can be stopped to allow West Mains Road traffic to turn right. Remember that all traffic is stopped while pedestrians and bicycles cross the junction. We encourage everyone to visit the junction and watch the sequence of lights carefully.

You must save a pdf of this workbook and upload it to Turn-it-In, by **16:00 on Thursday the 24th of October**. It must be uploaded together with a narrated video showing your simulation running. *Extensions are not permitted unless you have an approved learning adjustment.*

- 24 marks are avaiable for the design,
- 32 marks for the python programming,
- 24 marks for code documentation, and
- 20 marks for the video (including the commentary).

Please remember that programmes which exhibit dangerous behaviour (e.g. traffic and pedestrian lights on Go at the same time) will be an automatic fail.

Part 1 - Design (24 marks)

Firstly, I determined the length of time that each light would be on by looking at how long it took for the actual traffic lights to change. All of the yellow lights throughout the intersection have a 2 second lead time between red to green lights and a 3 second lead time before they are ready to change to red. Esslemount Road has a 35 second green light,Crossing has a 10 second green light. West Mains Road has a 20 second green light, and Mayfeild Road has a 30 second green light.And I added a button, when the pedestrian presses the button, the pedestrian's green light will light up after the traffic light cycle. On the other hand, if there are no pedestrians, then the cycle will always occur in the three roads.I set the space on the keyboard to be a pedestrian button press. When the button is pressed, it turns the cycle that was just a car road into a cycle that contains a green light for pedestrians.The safety of pedestrians and the speed at which the road passes are also ensured.

After making sure the Blinkstick was connected, I decided to define the colour of the LEDs first. I defined the colour constants for the LEDs as red, green, yellow, and off. By defining the function set_led_colors(colors), I set the corresponding colour on the BlinkStick for each LED. Using constants to define the colour code instead of repeating the RGB colour values in the code avoids the probability of errors and is easier to maintain.

To simplify traffic light status management, I used a dictionary STATES to define the color configuration of individual leds for each traffic status. By defining state configurations centrally, you can easily reference them in your program and reduce code duplication.

To keep the traffic lights repeating, I defined traffic_cycle, which contains the colour, description, and duration of each traffic light. Traffic lights are periodic changes in the arrangement of three lights, so using cycle() avoids using complex loops and makes each state repeat in turn.

Traffic lights need to work continuously on the road, so at the end of the program, I make the state of this traffic light cycle continuously. As for how to specify how long each colour of light works, I use a delay control to simulate the duration of the traffic light state. The corresponding LED colour configuration is then stored in STATES.

Part 2 - Python code (66 marks)

Write your code in the next cell. Please remember (for a B grade) we are looking for code which is an elegant model solution for the problem. It should demonstrate the use of the elements of Python needed to simulate the crossing as described above. Code should be clearly written, easy to read, make consistent use of spacing, be fully self-documenting and follow the Python naming conventions for constatnts, variables, functions, classes etc. Grades A3 to A1 require you to demonstrate learning that goes beyond what we have taught you and uses more advanced technqiues. If you choose to use such techniques, remember that code still needs to work propperly and be elegant. Overly complicated solutions could lose you marks!

- 32 marks for programming,
- 24 marks for code documentation

In [ ]:
```
import time
from blinkstick import blinkstick
from itertools import cycle
import keyboard
```

```
# find a blinkstick to use, if we don't find one raise an error.
bstick = blinkstick.find_first()
assert bstick is not None, "blinkstick not found."

# Define color constants
RED = "#460000" # red light
GREEN = "#00FF00" # green light
AMBER = "#201000" # amber light
OFF = "#000000" # no light


def set_led_colors(colors):
    index = 0
    while index < len(colors):
        current_color = colors[index]
        bstick.set_color(index=index, hex=current_color)
        index = index + 1

# Dictionary containing LED color configurations for each state
STATES = {
    "e_y": [RED, AMBER, OFF, RED, OFF, RED, OFF, RED],
    # ESSLEMOUNT is yellow. other are red.
    "e_g": [OFF, GREEN, OFF, RED, OFF, RED, OFF, RED],
    # ESSLEMOUNT is green.other are red.
    "e_a": [OFF, AMBER, OFF, RED, OFF, RED, OFF, RED],
    # ESSLEMOUNT is amber.other are red.
    "wm_y": [RED, OFF, OFF, RED, AMBER, RED, OFF, RED],
    # WEST_MAINS is yellow.other are red.
    "wm_g": [RED, OFF, OFF, RED, GREEN, OFF, OFF, RED],
    # WEST_MAINS is green.other are red.
    "wm_a": [RED, OFF, OFF, RED, AMBER, OFF, OFF, RED],
    # WEST_MAINS is amber.other are red.
    "m_y": [RED, OFF, OFF, RED, OFF, RED, AMBER, RED],
    # MAYFIELD is yellow.other are red.
    "m_g": [RED, OFF, OFF, RED, OFF, RED, GREEN, OFF],
    # MAYFIELD is green.other are red.
    "m_a": [RED, OFF, OFF, RED, OFF, RED, AMBER, OFF],
    # MAYFIELD is amber.other are red.
    "c_g": [RED, OFF, GREEN, OFF, OFF, RED, OFF, RED],
    # crossing is green. other are red.
    "all_red": [RED, OFF, OFF, RED, OFF, RED, OFF, RED]
    # all red
}

# Traffic light control process
# Define the cycle of traffic light changes with messages, states, and durations
car_cycle = [
    ("ESSLEMOUNT is yellow.", "e_y", 2),
    ("ESSLEMOUNT is green.", "e_g", 35),
    ("ESSLEMOUNT is amber.", "e_a", 3),
    ("WEST_MAINS is yellow.", "wm_y", 2),
    ("WEST_MAINS is green.", "wm_g", 20),
    ("WEST_MAINS is amber.", "wm_a", 3),
    ("MAYFIELD is yellow.", "m_y", 2),
    ("MAYFIELD is green.", "m_g", 30),
    ("MAYFIELD is amber.", "m_a", 3),
    ("All red.", "all_red", 1),
]

def ChangeFlag():
    global flag
    flag = True


flag = False
keyboard.add_hotkey("space", ChangeFlag)

# Main loop to control the traffic light cycle
while True:
    for message, state, duration in car_cycle:
        # Print the traffic message
        print(message, "Other roads are red.")
        # Set the LED colors according to the current state
        set_led_colors(STATES[state])
        # Wait for the specified duration
        time.sleep(duration)
    if flag:
        print("crossing is green.", "Other roads are red.")
        # Set the LED colors according to the current state
        set_led_colors(STATES["c_g"])
        # Wait for the specified duration
        time.sleep(10)
```

```
ESSLEMOUNT is yellow. Other roads are red.
ESSLEMOUNT is green. Other roads are red.
ESSLEMOUNT is amber. Other roads are red.
WEST_MAINS is yellow. Other roads are red.
WEST_MAINS is green. Other roads are red.
WEST_MAINS is amber. Other roads are red.
MAYFIELD is yellow. Other roads are red.
MAYFIELD is green. Other roads are red.
MAYFIELD is amber. Other roads are red.
All red. Other roads are red.
ESSLEMOUNT is yellow. Other roads are red.
ESSLEMOUNT is green. Other roads are red.
ESSLEMOUNT is amber. Other roads are red.
WEST_MAINS is yellow. Other roads are red.
WEST_MAINS is green. Other roads are red.
WEST_MAINS is amber. Other roads are red.
MAYFIELD is yellow. Other roads are red.
MAYFIELD is green. Other roads are red.
MAYFIELD is amber. Other roads are red.
All red. Other roads are red.
ESSLEMOUNT is yellow. Other roads are red.
ESSLEMOUNT is green. Other roads are red.
ESSLEMOUNT is amber. Other roads are red.
WEST_MAINS is yellow. Other roads are red.
WEST_MAINS is green. Other roads are red.
WEST_MAINS is amber. Other roads are red.
MAYFIELD is yellow. Other roads are red.
MAYFIELD is green. Other roads are red.
MAYFIELD is amber. Other roads are red.
All red. Other roads are red.
crossing is green. Other roads are red.
ESSLEMOUNT is yellow. Other roads are red.
ESSLEMOUNT is green. Other roads are red.
```

In [ ]:

```python
# turn off all the LEDs
for led in range(8):
    bstick.set_color(index=led, hex="#000000")
```

In [ ]:

```python

```