The project is implementing a smart traffic light management system, with the Python file acting as a sensor and a simulator of the cars moving through the junction, and the P4 file acting as the controller that determines which junction entrance should be green at what particular point in time.

The Python file is run in the format of `python traffic.py [add|quit] <junction1_car> <junction2_car> <junction3_car> <junction4_car>` whereby `<junctionX_car>` is the number of cars initially present at each entrance of the junction. At each iteration, the Python file also adds X cars at random times, using random.choices, with weights that can be chosen according to what the user desires. The number of cars to add X can be chosen according to what the user desires, to model the busyness of the junction, perhaps at different times of day.

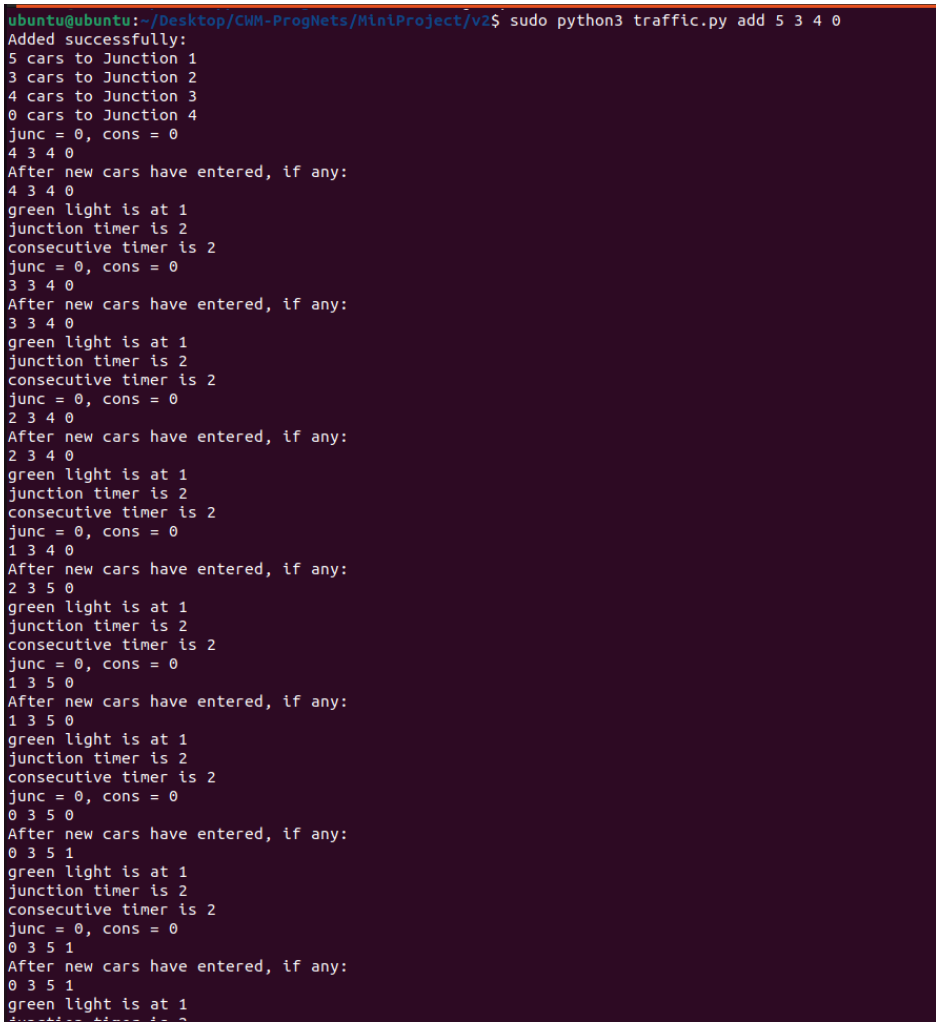
*Note: always run with `add`. `quit` does nothing useful except quit the programme as soon as it runs!

The project was done across multiple versions, each version building on top of the previous one and increasing in complexity. The following describes what each version does:

v1: Assume only one entrance to the junction exists. If a car is detected there, use the P4 file turn the lights green, and the Python file should simulate the car passing through the junction (by decrementing the number of cars at that entrance)

v2: Add support for multiple cars initially entering the junction. Still assume only one junction exists

An example result is shown below, for v2:




We can see a car is successfully randomly added to entrance 1 and 3 after the 4th iteration. And although the programme would usually decrement the number of cars at a green entrance to simulate the car passing through the junction, it successfully maintains the number of cars at that direction to be 0 instead of going to -1.

v3: Fix the functionality of the junction timer and consecutive timer. Above, it was stuck at 2, so we need to make sure it increments by 2 at each iteration.

```
ubuntu@ubuntu:~/Desktop/CWM-ProgNets/MiniProject/v3$ sudo python3 traffic.py add 5 3 4 0
Added successfully:
5 cars to Junction 1
3 cars to Junction 2
4 cars to Junction 3
0 cars to Junction 4
junc = 0, cons = 0
4 3 4 0
After new cars have entered, if any:
4 3 4 0
green light is at 1
junction timer is 0
consecutive timer is 0
end of loop


junc = 2, cons = 2
3 3 4 0
After new cars have entered, if any:
4 3 5 1
green light is at 1
junction timer is 2
consecutive timer is 2
end of loop


junc = 4, cons = 4
3 3 5 1
After new cars have entered, if any:
3 3 5 1
green light is at 1
junction timer is 4
consecutive timer is 4
end of loop


junc = 6, cons = 6
3 2 5 1
After new cars have entered, if any:
3 2 5 1
green light is at 2
junction timer is 6
consecutive timer is 6
end of loop
```

Successfully incremented both timers by 2.

v4: When new cars are consecutively detected at the green direction in intervals of at most 4s (i.e., 2 iterations), keep the light green. If no new cars are detected within this interval, then turn the light red and the next one green. This should be on for a maximum of 20 seconds (i.e., 10 iterations), at which it is forced to go red so that the next direction can be green.

```
junc = 2, cons = 2
2 0 4 0
After new cars have entered, if any:
2 0 5 0
green light is at 3
junction timer is 2
consecutive timer is 2
end of loop


junc = 4, cons = 0
2 0 4 0
After new cars have entered, if any:
2 0 4 0
green light is at 3
junction timer is 4
consecutive timer is 0
end of loop


junc = 6, cons = 2
2 0 3 0
After new cars have entered, if any:
2 0 3 0
green light is at 3
junction timer is 6
consecutive timer is 2
end of loop


junc = 8, cons = 4
2 0 2 0
After new cars have entered, if any:
2 0 2 1
green light is at 3
junction timer is 8
consecutive timer is 4
end of loop


junc = 0, cons = 0
2 0 2 0
After new cars have entered, if any:
2 0 2 0
green light is at 4
junction timer is 0
consecutive timer is 0
end of loop
```

An example for the 4 second consecutive interval is above. At the top iteration, we see that entrance 3 (which is the current green entrance) received a new car, so its consecutive timer resets to 0. However, after the fourth iteration, entrance 3 has not received any new cars, meanwhile its consecutive timer is already at 4. So we move on to turn entrance 4 green instead in the last iteration.

```
junc = 16, cons = 0
2 7 11 6
After new cars have entered, if any:
3 7 12 7
green light is at 1
junction timer is 16
consecutive timer is 0
end of loop


junc = 18, cons = 0
2 7 12 7
After new cars have entered, if any:
3 8 13 8
green light is at 1
junction timer is 18
consecutive timer is 0
end of loop


junc = 0, cons = 0
3 7 13 8
After new cars have entered, if any:
4 7 14 9
green light is at 2
junction timer is 0
consecutive timer is 0
end of loop
```

An example for the 20 second hard limit is above. It has been coded such that if junction timer were to be 20, it would reset itself to 0 on the same iteration and immediately move on to make the next entrance green.

v5: Add support for a “busy” operation mode. When an entrance is defined to be “busy”, which we will define as having 5 cars or more, then “rush” to make that entrance green. “Rush” means to only greenlight non-busy entrances for 8 seconds at max, until we get to the “busy” entrance which we will run for 20 seconds uninterruptedly. This was abandoned due to time constraints.

v6: A tidier version of v4. Added variables at the top of each code to easily change the code’s parameters when possible.
Here are the parameters in the Python file:

```
SLEEP_TIME = 0.5 # Amount of seconds to sleep in some of the sleep functions. Useful for reading command-line outputs
SECONDS_PER_ITERATION = 2 # Amount of seconds each iteration of the while loop should model
CARS_PER_ITERATION = 2 # Number of cars that can pass through the junction each iteration

# Percentage chance at each iteration this junction will get a new car incoming
J1_CHANCE = 30
J2_CHANCE = 70
J3_CHANCE = 60
J4_CHANCE = 50
```

And in the P4 file:

```
const bit<8> HARD_LIMIT = 20; // Maximum time a junction stays green
const bit<8> MAX_WAIT = 4; // Maximum interval between two cars approaching the
                           // green direction that the traffic light will wait for
```

Comments on both the P4 and Python files have also been added to help explain the flow of the code.

*Note: The final reformatting of the indentation was done on VS code and not on gedit in the lab machine. As of writing this, VS code does not detect any “inconsistent use of tab/whitespace” TabErrors. This should hopefully mean that no TabErrors are detected when the file is opened in the lab machine.